On the Expressiveness of Coordination

Thomas Given-Wilson, Axel Legay

On the Expressiveness of Coordination[☆]

Thomas Given-Wilson[a,*], Axel Legay[a]

[a]*Inria, Campus de Beaulieu, 263 Avenue du Général Leclerc, 35042 Rennes, France*

Abstract

The expressiveness of communication primitives has been explored in a common framework based on the π-calculus by considering four features: *synchronism* (asynchronous vs synchronous), *arity* (monadic vs polyadic data), *communication medium* (shared dataspaces vs channel-based), and *pattern-matching* (binding to a name vs testing name equality vs intensionality). Here another dimension *coordination* is considered that accounts for the number of processes required for an interaction to occur. Coordination generalises binary languages such as π-calculus to consider languages that can performing *joining* that combines inputs such as the Join Calculus, *splitting* that combines outputs; *full-coordination* that supports both joining and splitting. By means of possibility/impossibility of encodings, this paper shows the following results. Coordination is orthogonal to other features and no combination of non-coordination features can encode joining, splitting or full-coordination into a binary language. Joining and splitting can encode one another when the source language does not include name-matching or intensionality, and the target language can represent channel-based communication. Otherwise joining languages cannot encode splitting languages, and vice versa. Full-coordination cannot be encoded into either joining or splitting.

Keywords: Process Calculi, Expressiveness, Encodings, Pi-calculus, Join Calculus, Coordination

1. Introduction

The expressiveness of process calculi based upon their choice of communication primitives has been explored before [37, 8, 12, 24, 17, 19]. In [24] and [19] this is detailed by examining combinations of four features, namely: *synchronism*, asynchronous versus synchronous; *arity*, monadic versus polyadic; *communication medium*, shared dataspaces versus channels; and *pattern-matching*, purely binding names versus name equality versus intensionality. These features are able to represent many popular calculi [24, 19] such as: asynchronous or synchronous, monadic or polyadic π-calculus

[☆]An earlier version of this paper titled "On the Expressiveness of Joining" appeared in 8th Interaction and Concurrency Experience (ICE 2015)

[*]Corresponding author

Email addresses: thomas.given-wilson@inria.fr (Thomas Given-Wilson), axel.legay@inria.fr (Axel Legay)

[33, 34, 32]; LINDA [16]; Mobile Ambients [10]; μKLAIM [35]; semantic-π [11]; and asymmetric concurrent pattern calculus [18]. Also the intensional features capture significant aspects of Concurrent Pattern Calculus (CPC) [21, 22] and variations [17, 18]; and Psi calculi [1] and sorted Psi calculi [5].

Typically interaction in process calculi is a binary relation, where two processes interact and reduce to a third process. For example in π-calculus the interaction rule is

$$\overline{m}\langle a\rangle.P \mid m(x).Q \quad \longmapsto \quad P \mid \{a/x\}Q \ .$$

Here the processes $\overline{m}\langle a\rangle.P$ and $m(x).Q$ interact and reduce to a new process $P \mid \{a/x\}Q$. However, there are process calculi that are not binary with their interactions. For example, Concurrent Constraint Programming (CCP) has no direct interaction primitives, instead interactions are between a single process and the constraint environment [40]. In the other direction Join Calculus [15], general rendezvous calculus [3], and m-calculus [41] allow any number of processes to join in a single interaction. Recent work [23] considered some of these languages by adding an additional *coordination* feature that can be *binary* as above, or *joining* in the style of Join Calculus, general rendezvous calculus, and m-calculus. For a joining example, consider the reduction

$$\overline{m}\langle a\rangle.P_1 \mid \overline{n}\langle b\rangle.P_2 \mid [m(x) \mid n(y)] \rhd Q \quad \longmapsto \quad P_1 \mid P_2 \mid \{a/x, b/y\}Q$$

where the join $\rhd$ interacts when the two outputs $\overline{m}\langle a\rangle$ and $\overline{n}\langle b\rangle$ can match the two parts of the input $m(x)$ and $n(y)$, respectively.

This paper generalises the coordination of [23] to not only consider binary and joining interactions, but also *splitting* and *full-coordination*. Splitting is the dual of joining, where a single output can interact with multiple inputs, e.g.:

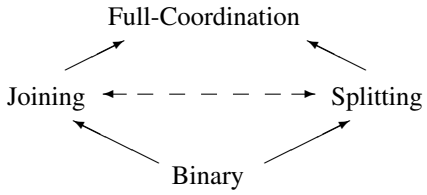
$$[\overline{m}\langle a\rangle \mid \overline{n}\langle b\rangle] \lhd P \mid m(x).Q_1 \mid n(y).Q_2 \quad \longmapsto \quad P \mid \{a/x\}Q_1 \mid \{b/y\}Q_2$$

where the split $[\overline{m}\langle a\rangle \mid \overline{n}\langle b\rangle]\lhd$ interacts when both inputs $m(x)$ and $n(y)$ can match the two parts of the split $\overline{m}\langle a\rangle$ and $\overline{n}\langle b\rangle$, respectively. Full-coordination is when both joining and splitting are present in the same language.

By extending the dimension of coordination, the original 48 calculi of [23] are here expanded to 96. This paper details the relations between various calculi focusing upon coordination with the following results.

Coordination is unrelated to synchronicity, arity, communication medium, or pattern-matching. In general this means that joining, splitting, or full-coordination do not support the encoding of one choice of a feature into another, unless this could already be achieved in the binary setting alone. For example, if there is no encoding from a synchronous binary language $\mathcal{L}_1$ into an asynchronous binary language $\mathcal{L}_2$, then there is no encoding from $\mathcal{L}_1$ into $\mathcal{L}_3$ where $\mathcal{L}_3$ is $\mathcal{L}_2$ with binary replaced by one of: joining, splitting, or full-coordination. That is, joining, splitting, or full-coordination languages cannot encode: synchronous communication into asynchronous communication; polyadic communication into monadic communication; channel-based communication into dataspace-based communication; nor intensionality into name-matching, or name-matching into non-matching unless some other combination of features could already do this. Similarly, no combination of the other features can encode joining or splitting into binary languages, or full-coordination into joining or splitting languages.

Within the coordination feature there are both orderings upon the languages groups, and some encodings between the groups. This is illustrated below where solid arrows are irreversible increases in expressiveness, and dashed arrows indicate partial equivalence results.



Both joining and splitting are strictly more expressive than binary communication, and both joining and splitting are strictly less expressive than full-coordination. However, there are encodings between joining and splitting languages that have channel-based communication and are no-matching (do not have name matching or intensionality). For example, synchronous polyadic channel-based no-matching joining and splitting languages can encode one another, indicating equivalent expressive power. However, encoding a joining (or splitting) language into a splitting (or joining, respectively) language is impossible if the target splitting (or joining) language does not have (or cannot support) channel-based communication. Similarly, if the source joining (or splitting) language has name matching (or intensionality) then encoding into a splitting (or joining, respectively) language is impossible.

The structure of the paper is as follows. Section 2 introduces the calculi considered here. Section 3 reviews the criteria used for comparing calculi. Section 4 considers encoding synchronism with coordination. Section 5 explores encoding arity via coordination. Section 6 presents results for encoding communication medium into coordination. Section 7 formalises that coordination cannot encode pattern-matching. Section 8 presents that coordination cannot be encoded by other features. Section 9 considers relations between different forms of coordination. Section 10 concludes, discusses future and related work, and provides some motivations for intensional calculi.

2. Calculi

This section defines the syntax, operational, and behavioural semantics of the calculi considered here. This relies heavily on the well-known notions developed for the π-calculus and adapts them when necessary to cope with different features. With the exception of the splitting and full-coordination details this repeats many prior definitions from [23], although there are minor syntactic changes for clarity in this work.

Assume a countable set of names $\mathcal{N}$ ranged over by $a, b, c, \ldots$. These form the foundation of two kinds of primitives involved in communication the *input patterns*

(denoted $p, q, \ldots$) and the *terms* (denoted $s, t, \ldots$); both are defined below.

$$\begin{array}{rcll} p, q & ::= & x & \textit{binding name} \\ & & \mid \ulcorner a \urcorner & \textit{name-match} \\ & & \mid p \bullet q & \textit{compound} \\ x \mid \ulcorner a \urcorner & =:: & m, n & \textit{name-match patterns} \\ s, t & ::= & a & \textit{name} \\ & & \mid s \bullet t & \textit{compound} \end{array}$$

The input patterns are used for input, with binding names doing binding, name-matches testing equality, and compounds supporting structure. The *name-match patterns* (denoted $m, n, \ldots$) are a subset of the input patterns that do not contain compounds. The terms are used for output, with names being the base and compounds adding structure. The free names and binding names input patterns and terms are as expected, taking the union of sub-patterns for compounds. Note that an input pattern is well-formed if and only if all binding names within the pattern are pairwise distinct. The rest of this paper will only consider well-formed input patterns.

The (parametric) syntax for the languages is:

$$\begin{array}{rcl} P, Q, R & ::= & \mathbf{0} \mid \mathit{OutProc} \mid \mathit{InProc} \mid (\nu a)P \mid P|Q \\ & & \mid \textbf{if } s = t \textbf{ then } P \textbf{ else } Q \mid *P \mid \surd . \end{array}$$

The different languages are obtained by replacing the output *OutProc* and input *InProc* with the various definitions. The rest of the process forms are as usual: $\mathbf{0}$ denotes the null process; restriction $(\nu a)P$ restricts the visibility of a to P; and parallel composition $P|Q$ allows independent evolution of P and Q. The **if** $s = t$ **then** P **else** Q represents conditional equivalence with **if** $s = t$ **then** P used when Q is $\mathbf{0}$. The $*P$ represents replication of the process P. Finally, the $\surd$ is used to represent a success process or state, exploited for reasoning about encodings as in [26, 17].

This paper considers the possible combinations of five features for communication: *synchronism* (asynchronous vs synchronous), *arity* (monadic vs polyadic data), *communication medium* (dataspace-based vs channel-based), *pattern-matching* (simple binding vs name equality vs intensionality), and *coordination* (binary vs joining vs splitting vs full-coordination). As a result there exist 96 languages denoted $\mathcal{L}_{\alpha,\beta,\gamma,\delta,\epsilon}$ where:

$\alpha =$ A for asynchronous communication, and S for synchronous communication.

$\beta =$ M for monadic data, and P for polyadic data.

$\gamma =$ D for dataspace-based communication, and C for channel-based communications.

$\delta =$ NO for no matching capability, NM for name-matching, and I for intensionality.

$\epsilon =$ B for binary communication, J for joining communication, S for splitting communication, and F for full-coordination communication.

$$
\begin{array}{rll}
\mathcal{L}_{-,-,-,-,B}: & \mathcal{I} ::= IN & O ::= OUT \\
\mathcal{L}_{-,-,-,-,J}: & \mathcal{I} ::= IN \mid \mathcal{I} \mid \mathcal{I} & O ::= OUT \\
\mathcal{L}_{-,-,-,-,S}: & \mathcal{I} ::= IN & O ::= OUT \mid O \mid O \\
\mathcal{L}_{-,-,-,-,F}: & \mathcal{I} ::= IN \mid \mathcal{I} \mid \mathcal{I} & O ::= OUT \mid O \mid O \\
\mathcal{L}_{A,-,-,-,-}: & InProc ::= [\mathcal{I}] \triangleright P & OutProc ::= [O] \triangleleft \\
\mathcal{L}_{S,-,-,-,-}: & InProc ::= [\mathcal{I}] \triangleright P & OutProc ::= [O] \triangleleft P \\
\mathcal{L}_{-,M,D,NO,-}: & IN ::= (x) & OUT ::= \langle a \rangle \\
\mathcal{L}_{-,M,D,NM,-}: & IN ::= (m) & OUT ::= \langle a \rangle \\
\mathcal{L}_{-,M,D,I,-}: & IN ::= (p) & OUT ::= \langle t \rangle \\
\mathcal{L}_{-,M,C,NO,-}: & IN ::= a(x) & OUT ::= \overline{a}\langle b \rangle \\
\mathcal{L}_{-,M,C,NM,-}: & IN ::= a(m) & OUT ::= \overline{a}\langle b \rangle \\
\mathcal{L}_{-,M,C,I,-}: & IN ::= s(p) & OUT ::= \overline{s}\langle t \rangle \\
\mathcal{L}_{-,P,D,NO,-}: & IN ::= (\widetilde{x}) & OUT ::= \langle \widetilde{a} \rangle \\
\mathcal{L}_{-,P,D,NM,-}: & IN ::= (\widetilde{m}) & OUT ::= \langle \widetilde{a} \rangle \\
\mathcal{L}_{-,P,D,I,-}: & IN ::= (\widetilde{p}) & OUT ::= \langle \widetilde{t} \rangle \\
\mathcal{L}_{-,P,C,NO,-}: & IN ::= a(\widetilde{x}) & OUT ::= \overline{a}\langle \widetilde{b} \rangle \\
\mathcal{L}_{-,P,C,NM,-}: & IN ::= a(\widetilde{m}) & OUT ::= \overline{a}\langle \widetilde{b} \rangle \\
\mathcal{L}_{-,P,C,I,-}: & IN ::= s(\widetilde{p}) & OUT ::= \overline{s}\langle \widetilde{t} \rangle
\end{array}
$$

Figure 1: Syntax of Languages.

For simplicity a dash $-$ will be used when the instantiation of a feature is unimportant.

Thus the syntax of every language is obtained from the productions in Figure 1. The denotation $\widetilde{\cdot}$ represents a sequence of the form $\cdot_1, \cdot_2, \ldots, \cdot_n$ and can be used for names, terms, and input patterns (also denote with $|\cdot|$ the size of a set, multiset, or sequence).

As usual $a(\ldots, x, \ldots) \triangleright P$ and $(\nu x)P$ and $(x \bullet \ldots) \triangleright P$ and $[\ldots \mid a(x) \mid \ldots] \triangleright P$ bind x in P. Observe that in $a(\ldots, \ulcorner b \urcorner, \ldots) \triangleright P$ and $(\ldots \bullet \ulcorner b \urcorner) \triangleright P$ neither a nor b bind in P, both are free. The corresponding notions of free and bound names of a process, denoted $\mathsf{fn}(P)$ and $\mathsf{bn}(P)$, are as usual. Also note that α-equivalence, denoted $=_\alpha$ is assumed in the usual manner. Further, an input is well-formed if all binding names in that input occur exactly once. This paper shall only consider well-formed inputs. Finally, the structural equivalence relation $\equiv$ is defined by:

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

$$\textbf{if } s = t \textbf{ then } P \textbf{ else } Q \equiv P \quad s = t \qquad \textbf{if } s = t \textbf{ then } P \textbf{ else } Q \equiv Q \quad s \neq t$$

$$P \equiv P' \quad \text{if } P =_\alpha P' \qquad (\nu a)\mathbf{0} \equiv \mathbf{0} \qquad (\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$$

$$P \mid (\nu a)Q \equiv (\nu a)(P \mid Q) \quad \text{if } a \notin \mathsf{fn}(P) \qquad * P \equiv P \mid * P \,.$$

Observe that $\mathcal{L}_{A,M,C,NO,B}$, $\mathcal{L}_{A,P,C,NO,B}$, $\mathcal{L}_{S,M,C,NO,B}$, and $\mathcal{L}_{S,P,C,NO,B}$ align with the communication primitives of the asynchronous/synchronous monadic/polyadic π-calculus

[33, 34, 32]. The language $\mathcal{L}_{A,P,D,NM,B}$ aligns with LINDA[16]; the languages $\mathcal{L}_{A,M,D,NO,B}$ and $\mathcal{L}_{A,P,D,NO,B}$ with the monadic/polyadic Mobile Ambients [10]; and $\mathcal{L}_{A,P,C,NM,B}$ with that of μKLAIM [35] or semantic-π [11].

The intensional languages do not in general exactly match any well-known calculi. However, the language $\mathcal{L}_{S,M,D,I,B}$ is the asymmetric concurrent pattern calculus of [18] and similar calculi have been mentioned in [17], as variations of Concurrent Pattern Calculus [21, 17] with their behavioural theory as a specialisation of [20]. Similarly, the language $\mathcal{L}_{S,M,C,I,B}$ is very similar to pattern-matching Spi calculus [27] and Psi calculi [1], albeit without the assertions or the possibility of repeated binding names in patterns. There are also similarities between $\mathcal{L}_{S,M,C,I,B}$ and the polyadic synchronous π-calculus of [9], although the intensionality is limited to the channel, i.e. inputs and outputs of the form $s(x).P$ and $\overline{s}\langle a\rangle.P$ respectively.

For the joining languages: $\mathcal{L}_{A,P,C,NO,J}$ represents Join Calculus [15]; and $\mathcal{L}_{S,P,C,NO,J}$ the general rendezvous calculus [3], and m-calculus [41], although the latter has higher order constructs and other aspects that are not captured within the features here. There are no exact connections for the splitting languages. However, $\mathcal{L}_{S,M,C,NO,S}$ is related to broadcast calculus [39] and $b\pi$-calculus [14] although they place side conditions on the splitting outputs. Finally, the full-coordination languages have no obvious connections to existing languages.

Remark 2.1. *The languages here can be easily ordered; in particular $\mathcal{L}_{\alpha_1,\beta_1,\gamma_1,\delta_1,\epsilon_1}$ can be encoded into $\mathcal{L}_{\alpha_2,\beta_2,\gamma_2,\delta_2,\epsilon_2}$ if it holds that $\alpha_1 \leq \alpha_2$ and $\beta_1 \leq \beta_2$ and $\gamma_1 \leq \gamma_2$ and $\delta_1 \leq \delta_2$ and $\epsilon_1 \leq \epsilon_2$, where $\leq$ is the least reflexive relation satisfying the following axioms:*

$$A \leq S \qquad M \leq P \qquad D \leq C \qquad NO \leq NM \leq I \qquad B \leq J \leq F \quad B \leq S \leq F\ .$$

This can be understood as a limited language variation being a special case of a more general language. Asynchronous communication is synchronous communication with all output followed by $\mathbf{0}$*. Monadic communication is polyadic communication with all tuples of arity one. Dataspace-based communication is channel-based communication with all k-ary tuples communicating with channel name k. All name-matching communication is intensional communication without any compounds, and no-matching capability communication is both without any compounds and with only binding names in patterns. Lastly, binary communication is: joining communication with all joining inputs having only a single input pattern, and splitting communication with all splitting outputs having only a single output term. Joining is full-coordination with splitting outputs having only a single output term, and splitting is full-coordination with joining inputs having only a single input pattern.*

The operational semantics of the languages is given here via reductions as in [32, 29, 19]. An alternative style is via a *labelled transition system* (LTS) such as [24]. Here the reduction based style is to simplify having to define here the (potentially complex) labels that occur when both intensionality, and joining/splitting/full-coordination is in play. The LTS style can be used for intensional languages [1, 17, 20]. Also, for the non-binary languages the techniques used in [4] can be used directly for the no-matching

joining languages, with the techniques of [4, 20] used to extend intensionality and other coordination forms[1].

Substitutions (denoted $\sigma, \rho, \ldots$) in non-pattern-matching and name-matching languages are mappings from names to names. For intensional languages substitutions are mappings from names to terms. Note that substitutions are assumed to have finite domain. The application of a substitution σ to a pattern p is defined as follows:

$$\sigma x = \sigma(x) \;\; x \in \mathrm{domain}(\sigma) \qquad \sigma x = x \;\; x \notin \mathrm{domain}(\sigma)$$
$$\sigma\ulcorner x \urcorner = \ulcorner(\sigma x)\urcorner \qquad \sigma(p \bullet q) = (\sigma p) \bullet (\sigma q)\ .$$

Where substitution is as usual on names, and on the understanding that the name-match syntax can be applied to any term as follows $\ulcorner x \urcorner \stackrel{\text{def}}{=} \ulcorner x \urcorner$ and $\ulcorner(s \bullet t)\urcorner \stackrel{\text{def}}{=} \ulcorner s \urcorner \bullet \ulcorner t \urcorner$.

Given a substitution σ and a process P, denote with σP the (capture-avoiding) application of σ to P that behaves in the usual manner. Note that capture can always be avoided by exploiting α-equivalence, which can in turn be assumed [42, 2].

Interaction between processes is handled in two parts: the matching of terms, and input patterns; and the synchronisation of various inputs and outputs.

The matching of terms $\widetilde{t}$ with some patterns $\widetilde{p}$ is handled in two parts. First, the *match* rule $\{t /\!\!/ p\}$ of a single term t with a single pattern p to create a substitution σ:

$$\{t /\!\!/ x\} \stackrel{\text{def}}{=} \{t/x\} \qquad \{s \bullet t /\!\!/ p \bullet q\} \stackrel{\text{def}}{=} \{s /\!\!/ p\} \cup \{t /\!\!/ q\}$$
$$\{a /\!\!/ \ulcorner a \urcorner\} \stackrel{\text{def}}{=} \{\} \qquad \{t /\!\!/ p\} \;\; \text{undefined} \;\; \text{otherwise.}$$

Any term t can be matched with a binding name x to generate a substitution from the binding name to the term $\{t/x\}$. A single name a can be matched with a name-match for that name $\ulcorner a \urcorner$ to yield the empty substitution. A compound term $s \bullet t$ can be matched by a compound pattern $p \bullet q$ when the components match to yield substitutions $\{s /\!\!/ p\} = \sigma_1$ and $\{t /\!\!/ q\} = \sigma_2$, the resulting substitution is the unification of σ_1 and σ_2. Observe that since patterns are well-formed, the substitutions of components will always have disjoint domain. Otherwise the match is undefined.

The second part is then the *poly-match* rule $\textsc{Match}(\widetilde{t}; \widetilde{p})$ that determines matching of a sequence of terms $\widetilde{t}$ with a sequence of patterns $\widetilde{p}$, that is defined below.

$$\textsc{Match}(;) = \{\} \qquad \frac{\{s /\!\!/ p\} = \sigma_1 \qquad \textsc{Match}(\widetilde{t}; \widetilde{q}) = \sigma_2}{\textsc{Match}(s, \widetilde{t}; p, \widetilde{q}) = \sigma_1 \cup \sigma_2}\ .$$

The empty sequence matches with the empty sequence to produce the empty substitution. Otherwise when there is a sequence of terms $s, \widetilde{t}$ and a sequence of patterns $p, \widetilde{q}$, the first elements are matched $\{s /\!\!/ p\}$ and the remaining sequences use the poly-match rule. If both are defined and yield substitutions, then the union of substitutions is the result. (Like the match rule, the union is ensured disjoint domain by well-formedness of inputs.) Otherwise the poly-match rule is undefined, for example when a single match fails, or the sequences are of unequal arity.

[1] Although not yet proven, this should be straightforward albeit very tedious.

To handle the more complex constraints for full-coordination languages define the *synchronisation* $\textsc{Sync}(\widetilde{\overline{sm}\langle t\rangle}, \widetilde{sn(p)}) = \widetilde{\sigma}$ that takes a multiset of outputs $\overline{sm}\langle t\rangle$ and a sequence of inputs $sn(p)$ and produces a sequence of substitutions σ as follows:

$$\textsc{Sync}(;) = \{\} \qquad \frac{\textsc{Match}(t_1; p_1) = \sigma_1 \qquad \textsc{Sync}(\ \widetilde{\overline{sm}\langle t\rangle}\backslash\{\overline{s}\langle t_1\rangle\}\ ,\ \widetilde{sn(p)}\ ) = \widetilde{\sigma}}{\textsc{Sync}(\ \widetilde{\overline{sm}\langle t\rangle}\ ,\ s(p_1); \widetilde{sn(p)}\ ) = \sigma_1; \widetilde{\sigma}}\ .$$

That is, the synchronisation of an empty multiset and an empty sequence is an empty sequence. Otherwise, if the first input $s(p_1)$ in the sequence can be matched against one of the outputs $\overline{s}\langle t_1\rangle$ to yield a substitution σ_1, and the remaining outputs and inputs can by synchronised to produce a sequence of substitutions $\widetilde{\sigma}$, then append $\widetilde{\sigma}$ to σ_1. Otherwise the sycnhronisation is undefined, for example when one input cannot be matched against any output, or the multiset and sequence are of different sizes. Thus the synchronisation produces a sequence of substitutions that correspond to the inputs. Note that synchronisations are defined by existence and not any particular choice of matching output and input. For example, consider: $\textsc{Sync}(\{\overline{a}\langle a\rangle, \overline{a}\langle b\rangle\}, a(x), a(\ulcorner a \urcorner))$, although a first step $\{a/x\}; \textsc{Sync}(\{\overline{a}\langle b\rangle\}, a(\ulcorner a \urcorner))$ may appear to lead to failure (since $\textsc{Match}(b, \ulcorner a \urcorner)$ is undefined), a different first step $\{b/x\}; \textsc{Sync}(\{\overline{a}\langle a\rangle\}, a(\ulcorner a \urcorner))$ yields $\{b/x\}; \{\}$.

Now the reduction for all the languages is defined by:

$$\frac{\begin{array}{c}\widetilde{\rho} = \textsc{Sync}(\{\widetilde{\overline{sm_1}\langle t_1\rangle}\} \cup \ldots \cup \{\widetilde{\overline{sm_i}\langle t_i\rangle}\}; \widetilde{sn_1(p_1)}, \ldots, \widetilde{sn_j(p_j)}) \\ \sigma_1 = \rho_1 \cup \ldots \cup \rho_k \quad k = |\widetilde{sn_1(p_1)}| \\ \sigma_i = \rho_{n+1} \cup \ldots \cup \rho_{n+k} \quad n = \Sigma_{x\in\{1,\ldots,j-1\}}|\widetilde{sn_x(p_x)}| \quad k = |\widetilde{sn_i(p_i)}|\end{array}}{\begin{array}{r}[\widetilde{\overline{sm_1}\langle t_1\rangle}] \lhd P_1 \mid \ldots \mid [\widetilde{\overline{sm_i}\langle t_i\rangle}] \lhd P_i \mid [\widetilde{sn_1(p_1)}] \rhd Q_1 \mid \ldots \mid [\widetilde{sn_j(p_j)}] \rhd Q_j \\ \longmapsto P_1 \mid \ldots \mid P_i \mid \sigma_1 Q_1 \mid \ldots \mid \sigma_j Q_j\end{array}}$$

That is, when there are several splits and joins and all the outputs can be synchronised with all the inputs, then reduce to the (P's from the splits in the synchronous setting in parallel with the) substitutions constructed from the synchronisation applied to the appropriate Q's. The only non-trivial part is the definition of each substitution σ_i for the inputs $\widetilde{sn_i(p_i)}$, which is the union of the substitutions $\widetilde{\rho}$ that correspond to the inputs, taken by their position in the sequence generated by the synchronisation rule.

The definition of the synchronisation rule and main reduction rule above is necessary for the full-coordination languages, but not necessarily for binary, joining, or splitting languages. Indeed, all except full-coordination can be defined with simpler rules. However, this makes the reasoning later more arduous since delicacy is then required to clearly denote which rule is being applied for which language, and general results that reason over multiple forms of coordination need to have separate cases for each possible base reduction rule. The choice here is to use a single rule since this simplifies the later results significantly, at the cost of a little extra weight here.

The general reduction relation $\longmapsto$ also includes the following three rules:

$$\frac{P \longmapsto P'}{P \mid Q \longmapsto P' \mid Q} \qquad \frac{P \longmapsto P'}{(\nu a)P \longmapsto (\nu a)P'} \qquad \frac{P \equiv Q \quad Q \longmapsto Q' \quad Q' \equiv P'}{P \longmapsto P'}\ .$$

The reflexive transitive closure of $\longmapsto$ is denoted by $\Longmapsto$.

Lastly, for each language let $\simeq$ denote a reduction-sensitive reference behavioural equivalence for that language, e.g. a barbed equivalence. For the non-intensional languages these are mostly already known, either by their equivalent language in the literature, such as asynchronous/synchronous monadic/polyadic π-calculus or join calculus, or from [24]. For the intensional languages the results in [20] can be used. For the joining languages that reflect those of the literature the techniques used in [4] apply. For other combinations of joining, splitting, and full-coordination; as well as the addition of intensionality to non-binary languages, adaptations of [4, 20] should prove adequate[2].

3. Encodings

This section recalls the definition of valid encodings as well as some useful theorems (details in [26]) for formally relating process calculi.

The choice of valid encodings here is that used, sometimes with mild adaptations, in [26, 25, 21, 36, 17, 22] and has also inspired similar works [30, 31, 43]. However, there are alternative approaches to encoding criteria or comparing expressive power [6, 13, 9, 38, 43]. Further arguments for, and against, the valid encodings here can be found in [26, 25, 43, 22].

An *encoding* of a language $\mathcal{L}_1$ into another language $\mathcal{L}_2$ is a pair $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\rrbracket})$ where $\llbracket\cdot\rrbracket$ translates every $\mathcal{L}_1$-process into an $\mathcal{L}_2$-process and $\varphi_{\llbracket\rrbracket}$ maps every name (of the source language) into a tuple of k names (of the target language), for $k > 0$. The translation $\llbracket\cdot\rrbracket$ turns every term of the source language into a term of the target; in doing this, the translation may fix some names to play a precise rôle or may translate a single name into a tuple of names. This can be obtained by exploiting $\varphi_{\llbracket\rrbracket}$.

Now consider only encodings that satisfy the following properties. Let a *k-ary context* $C(\cdot_1;\ldots;\cdot_k)$ be a term where k occurrences of $\mathbf{0}$ are linearly replaced by the holes $\{\cdot_1;\ldots;\cdot_k\}$ (every one of the k holes must occur once and only once). Denote with $\longmapsto_i$ and $\Longmapsto_i$ the relations $\longmapsto$ and $\Longmapsto$ in language $\mathcal{L}_i$; denote with $\longmapsto_i^\omega$ an infinite sequence of reductions in $\mathcal{L}_i$. Moreover, let $\simeq_i$ denote the reference behavioural equivalence for language $\mathcal{L}_i$. Also, let $P \Downarrow_i$ mean that there exists P' such that $P \Longmapsto_i P'$ and $P' \equiv P'' \mid \surd$, for some P''. Finally, to simplify reading, let S range over processes of the source language (viz., $\mathcal{L}_1$) and T range over processes of the target language (viz., $\mathcal{L}_2$).

Valid Encoding An encoding $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\rrbracket})$ of $\mathcal{L}_1$ into $\mathcal{L}_2$ is *valid* if it satisfies the following five properties:

1. *Compositionality:* for every k-ary operator op of $\mathcal{L}_1$ and for every subset of names N, there exists a k-ary context $C^N_{\mathsf{op}}(\cdot_1;\ldots;\cdot_k)$ of $\mathcal{L}_2$ such that, for all $S_1,\ldots,S_k$ with $\mathsf{fn}(S_1,\ldots,S_k) = N$, it holds that $\llbracket\, \mathsf{op}(S_1,\ldots,S_k)\,\rrbracket = C^N_{\mathsf{op}}(\llbracket\, S_1\,\rrbracket;\ldots;\llbracket\, S_k\,\rrbracket)$.

[2]This has not been proven as yet, however there appears no reason it should not be possible, and the results here rely upon the existence of an equivalence relation, not any particular one.

2. *Name invariance:* for every S and substitution σ, it holds that $[\![\sigma S]\!] = \sigma'[\![S]\!]$ if σ is injective and $[\![\sigma S]\!] \simeq_2 \sigma'[\![S]\!]$ otherwise where σ' is such that $\varphi_{[\![ ]\!]}(\sigma(a)) = \sigma'(\varphi_{[\![ ]\!]}(a))$ for every name a.
3. *Operational correspondence:*
   - for all $S \Longmapsto_1 S'$, it holds that $[\![S]\!] \Longmapsto_2 \simeq_2 [\![S']\!]$;
   - for all $[\![S]\!] \Longmapsto_2 T$, there exists S' such that $S \Longmapsto_1 S'$ and $T \Longmapsto_2 \simeq_2 [\![S']\!]$.
4. *Divergence reflection:* for every S such that $[\![S]\!] \longmapsto_2^\omega$, it holds that $S \longmapsto_1^\omega$.
5. *Success sensitiveness:* for every S, it holds that $S \Downarrow_1$ if and only if $[\![S]\!] \Downarrow_2$.

Now recall two results concerning valid encodings that are useful for later proofs.

Proposition 3.1 (Proposition 5.5 from [26])**.** *Let $[\![\cdot]\!]$ be a valid encoding; then, $S \not\longmapsto_1$ implies that $[\![S]\!] \not\longmapsto_2$.*

Proposition 3.2 (Proposition 5.6 from [26])**.** *Let $[\![\cdot]\!]$ be a valid encoding; then for every set of names N, it holds that $C_{|}^{N}(\cdot_1, \cdot_2)$ has both its holes at top-level.*

4. Coordination and Synchronicity

This section considers the relation between coordination and synchronicity. it turns out that coordination is unable to encode synchronicity unless it could otherwise be encoded by other features.

In general synchronous communication can be encoded into asynchronous communication when the target language includes: channel names; name-matching and polyadicity; or intensionality. Thus it is sufficient to consider the languages $\mathcal{L}_{A,M,D,NO,-}$ and $\mathcal{L}_{A,P,D,NO,-}$ and $\mathcal{L}_{A,M,D,NM,-}$ since the other asynchronous languages can encode their synchronous joining counterparts in the usual manner [28, 7]. For example, the encoding from $\mathcal{L}_{S,M,C,NO,B}$ into $\mathcal{L}_{A,M,C,NO,B}$ given by

$$
\begin{array}{rcl}
[\![[\overline{n}\langle a\rangle] \triangleleft P]\!] & \stackrel{\text{def}}{=} & (\nu z)([\overline{n}\langle z\rangle] \triangleleft \mid [z(x)] \triangleright ([\overline{x}\langle a\rangle] \triangleleft \mid [\![P]\!])) \\
[\![[n(a)] \triangleright Q]\!] & \stackrel{\text{def}}{=} & (\nu x)[n(z)] \triangleright ([\overline{z}\langle x\rangle] \triangleleft \mid [x(a)] \triangleright [\![Q]\!])
\end{array}
$$

can be adapted in the obvious manner for $\mathcal{L}_{S,M,C,NO,J}$ into $\mathcal{L}_{A,M,C,NO,J}$ as follows

$$
\begin{array}{rcl}
[\![[\overline{n}\langle a\rangle] \triangleright P]\!] & \stackrel{\text{def}}{=} & (\nu z)([\overline{n}\langle z\rangle] \triangleleft \mid [z(x)] \triangleright ([\overline{x}\langle a\rangle] \triangleleft \mid [\![P]\!])) \\
[\![[n_1(a_1) \mid \ldots \mid n_i(a_i)] \triangleright Q]\!] & \stackrel{\text{def}}{=} & (\nu x_1, \ldots, x_i)[n_1(z_1) \mid \ldots \mid n_i(z_i)] \triangleright \\
 & & \quad ([\overline{z_1}\langle x_1\rangle] \triangleleft \mid \ldots \mid [\overline{z_i}\langle x_i\rangle] \triangleleft \\
 & & \quad\quad \mid [x_1(a_1) \mid \ldots \mid x_i(a_i)] \triangleright [\![Q]\!]) \, .
\end{array}
$$

The idea for binary languages is that the encoded output creates a fresh name z and sends it to the encoded input. The encoded input creates a fresh name x and sends it to the encoded output along channel name z. The encoded output now knows it has communicated and evolves to $[\![P]\!]$ in parallel with the original a sent to the encoded input along channel name x. When the encoded input receives this it can evolve to

$[\![Q]\!]$. The joining version is similar except the join synchronises with all the encoded outputs at once, sends the fresh names x_j in parallel, and then synchronises on all the a_j in the last step.

The encoding above is shown for $\mathcal{L}_{S,M,C,NO,J}$ into $\mathcal{L}_{A,M,C,NO,J}$ and is the identity on all other process forms. This can be proven to be a valid encoding.

Lemma 4.1. *Given a $\mathcal{L}_{S,M,C,NO,J}$ input P and output Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by induction on the number of input patterns in P and then for each one by definition of the poly-match rule. □

Lemma 4.2. *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. The proof is trivial for all the primitives except input and output as they are translated homomorphically. The output is straightforward, the input is by induction on the number of inputs in the join. □

Lemma 4.3. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{S,M,C,NO,J}$ into $\mathcal{L}_{A,M,C,NO,J}$ preserves and reflects reductions.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 4.1, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The size of k in both cases is $2 + i$ where i is the number of input-patterns of the input involved in $P \longmapsto P'$. The inductive cases where the last rule used is a structural one then rely on Lemma 4.2. □

Theorem 4.4. *There is a valid encoding from $\mathcal{L}_{S,M,C,NO,J}$ into $\mathcal{L}_{A,M,C,NO,J}$.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 4.3. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 4.3 k times and Lemma 4.2 obtain that $[\![P]\!] \longmapsto^j [\![P']\!] \equiv [\![P'']\!] \mid \surd$ where j can be determined from the instantiations of Lemma 4.2, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. □

Splitting can be adapted in a similar manner, e.g. consider for $\mathcal{L}_{S,M,C,NO,S}$ into $\mathcal{L}_{A,M,C,NO,S}$

$$
\begin{aligned}
[\![\, [\overline{n_1}\langle a_1\rangle \mid \ldots \mid \overline{n_i}\langle a_i\rangle] \lhd P \,]\!] &\stackrel{\text{def}}{=} (\nu z_1, \ldots, z_i)([\overline{n_1}\langle z_1\rangle \mid \ldots \mid \overline{n_1}\langle z_i\rangle] \lhd \mid \\
&\qquad [z_1(x_1)] \rhd \ldots \rhd [z_i(x_i)] \rhd \\
&\qquad\quad ([\overline{x_1}\langle a_1\rangle \mid \ldots \mid \overline{x_i}\langle a_i\rangle] \lhd \mid [\![\, P \,]\!])) \\
[\![\, [a(b)] \rhd Q \,]\!] &\stackrel{\text{def}}{=} (\nu x)[a(z)] \rhd ([\overline{z}\langle x\rangle] \lhd \mid [x(b)] \rhd [\![\, Q \,]\!])
\end{aligned}
$$

The use of fresh names z and x is as before. The splitting version is similar except the split synchronises with all the encoded inputs at once, sending fresh names z_j in

parallel, then collects all the responses with fresh names x_j, and then splits sending all the original names a_i at once in the last step.

The encoding above for $\mathcal{L}_{S,M,C,NO,S}$ into $\mathcal{L}_{A,M,C,NO,S}$ is the identity on all other process forms. This can similarly be proven to be a valid encoding.

Theorem 4.5. *There is a valid encoding from $\mathcal{L}_{S,M,C,NO,S}$ into $\mathcal{L}_{A,M,C,NO,S}$.*

Proof. The proof is almost identical to Theorem 4.4, the only changes are straighforward adaptations of Lemma 4.1 and Lemma 4.2 (to be by induction on the number of outputs rather than inputs). □

The same can be shown for the encoding from $\mathcal{L}_{S,M,C,NO,F}$ into $\mathcal{L}_{A,M,C,NO,F}$ by taking the encoding of inputs from the joining and outputs from the splitting encodings in the obvious manner.

Theorem 4.6. *There is a valid encoding from $\mathcal{L}_{S,M,C,NO,F}$ into $\mathcal{L}_{A,M,C,NO,F}$.*

Proof. The proof is the straightforward combination of the techniques for Theorems 4.4 & 4.5. □

Corollary 4.7. *If there exists a valid encoding from $\mathcal{L}_{S,\beta,\gamma,\delta,B}$ into $\mathcal{L}_{A,\beta,\gamma,\delta,B}$ then there exists a valid encoding from $\mathcal{L}_{S,\beta,\gamma,\delta,\epsilon}$ into $\mathcal{L}_{A,\beta,\gamma,\delta,\epsilon}$.*

Proof. Theorems 4.4, 4.5, & 4.6 provide the foundation for all the channel-based results. The only other cases can encode channels and so use encodings of the channel-based solution above. For the polyadic and name-matching languages this holds by Proposition 4.1 of [24], otherwise for the intensional languages this holds by Theorem 6.4 of [19]. □

These results confirm that the ability to encode synchronous communication into asynchronous communication is not impacted by changes to coordination. Any encoding that holds between binary languages also holds for the corresponding languages with other coordination forms. Thus no expressiveness is lost by changing from binary languages to other coordination forms, and existing results can easily be transferred.

The following results formalise that no new encodings or expressiveness is gained within joining, splitting, or full-coordination languages due to the shift from binary languages. The impossibility of encoding $\mathcal{L}_{S,M,D,NM,J}$ into $\mathcal{L}_{A,M,D,NM,J}$ is detailed as it illustrates the key proof technique. The other results are either simpler variations (i.e. without name-matching) or straightforward adaptations to consider splitting or full-coordination.

Theorem 4.8. *There exists no valid encoding from $\mathcal{L}_{S,M,D,NM,J}$ into $\mathcal{L}_{A,M,D,NM,J}$.*

Proof. The proof is by contradiction. Consider two processes $P = [(x)] \triangleright \textbf{if } x = b \textbf{ then } \Omega$ (where Ω is a divergent process) and $Q = [\langle a \rangle] \triangleleft Q'$. Since $P \mid Q \longmapsto$ by validity of the encoding $[\![P \mid Q]\!] \longmapsto$ and this must be between some $R_1 = [\langle m \rangle] \triangleleft$ for some m and R_2. Observe that $R_1 \mid R_2$ cannot be a reduct of either $[\![P]\!]$ or $[\![Q]\!]$ since then either P or Q would reduce and this contradicts Proposition 3.1.

If R_1 arises from $[\![P]\!]$ then it can be shown that $[\![P]\!]$ must also include a top level join since otherwise there would be no join in $[\![P]\!]$ that can bind some name to x and

name invariance or divergence reflection would be shown to fail (i.e. $P \mid Q \longmapsto \textbf{if } a = b \textbf{ then } \Omega \mid Q'$ and $\{b/a\}\textbf{if } a = b \textbf{ then } \Omega \mid Q' \longmapsto^{\omega}$ while $C_{\mid}^{\mathcal{N}}([\![P]\!], [\![Q]\!]) \Longmapsto$ does no inputs on any part of $[\![P]\!]$ and so must always or never diverge regardless of interaction with $[\![Q]\!]$). Thus $[\![P]\!]$ must include a top level join and further it must include an input pattern $(\ulcorner n \urcorner)$ for some $n \neq m$ since otherwise if the join was only $[(z_1) \mid \ldots \mid (z_i)] \triangleright R'$ for some $\widetilde{z}$ and R' then $[\![P \mid \ldots \mid P]\!]$ for i instances of P would reduce while $P \mid \ldots \mid P$ does not contradicting Proposition 3.1. It follows that $[\![Q]\!]$ must include $(\ulcorner m \urcorner)$ as part of some join under which there must be an output that is able to send at least one name to $[\![P]\!]$ via an output $\langle d \rangle$ for some d (this could be any number of names, but assume 1 here for simplicity). Now consider the name d.

- If $d = m$ then $[\![P]\!] \longmapsto$ and this contradicts validity of the encoding since $P \not\longmapsto$.
- If $d = n$ then n is not bound in $[\![P]\!]$ and so it can be shown that either: this fails name invariance or divergence reflection (again by $P \mid Q \longmapsto \textbf{if } a = b \textbf{ then } \Omega \mid Q'$ and $\{b/a\}\textbf{if } a = b \textbf{ then } \Omega \mid Q' \longmapsto^{\omega}$); or there must be a further input in $[\![P]\!]$ that is binding as in the next case.
- If $d \neq m$ and $d \neq n$ then it can be shown that $[\![P \mid Q \mid P]\!]$ can reduce such that the input under consideration interacts with the $\langle m \rangle$ from the other $[\![P]\!]$ and this ends up contradicting operational correspondence.

If R_1 arises from $[\![Q]\!]$ then it can be shown that $[\![Q]\!]$ must also include a top level join since otherwise when $Q' = \Omega$ then $[\![Q]\!]$ would always diverge or never diverge regardless of interaction with $[\![P]\!]$ and this contradicts divergence reflection. Thus $[\![Q]\!]$ must include a top level join and further it must include an input pattern $(\ulcorner n \urcorner)$ for some $n \neq m$ since otherwise if the join was only $[(z_1) \mid \ldots \mid (z_i)] \triangleright R'$ for some $\widetilde{z}$ and R' then $[\![Q \mid \ldots \mid Q]\!]$ for i instances of Q would reduce while $Q \mid \ldots \mid Q$ does not contradicting Proposition 3.1. Now consider when $Q' = \textbf{if } a = b \textbf{ then } \surd$ and the substitution $\sigma = \{b/a\}$. Clearly $P \mid \sigma Q \mid Q \longmapsto S$ where either: $S \longmapsto^{\omega}$ and $S \Downarrow$; or $S \not\longmapsto^{\omega}$ and $S \not\Downarrow$. However it can be shown that the top level join in $[\![Q]\!]$ is not able to discriminate and thus that there exist two possible reductions $[\![P \mid \sigma Q \mid Q]\!] \longmapsto R'$ to an R' where either: $R' \longmapsto^{\omega}$ and $R' \not\Downarrow$; or $R' \not\longmapsto^{\omega}$ and $R \Downarrow$; both of which contradict divergence reflection and success sensitiveness. □

Theorem 4.9. *There exists no valid encoding from $\mathcal{L}_{S,M,D,NM,S}$ into $\mathcal{L}_{A,M,D,NM,S}$.*

Proof. This is proved in a very similar manner to Theorem 4.8. □

Corollary 4.10. *If there exists no valid encoding from $\mathcal{L}_{S,\beta_1,\gamma_1,\delta_1,B}$ into $\mathcal{L}_{A,\beta_2,\gamma_2,\delta_2,B}$, then there exists no valid encoding from $\mathcal{L}_{S,\beta_1,\gamma_1,\delta_1,\epsilon}$ into $\mathcal{L}_{A,\beta_2,\gamma_2,\delta_2,\epsilon}$.*

Proof. The techniques in Theorems 4.8 & 4.9 apply to all monadic joining and splitting languages, respectively. Monadic no-matching languages are simpler variants of the same proof technique, while polyadic no-matching (since polyadic name-matching can encode synchronous communication into asynchronous) is a simple generalisation of the above proofs. Joining and splitting proofs can be combined for the full-coordination languages. □

That joining, splitting, or full-coordination do not allow for an encoding of synchronous communication alone is not surprising, since there is no control in the input of which outputs are interacted with (without some other control such as channel names or pattern-matching). Thus, being able to consume more outputs or inputs in a single interaction does not capture synchronous behaviours.

This formalises that there is no change to results within languages grouped by their coordination form. Separation results between coordination forms, and thus the conclusion that synchronism and coordination are orthogonal are concluded in Section 8.

5. Coordination and Arity

This section considers the relation between non-binary coordination and arity. Although there appear to be some similarities in that both have a base case (monadic or binary), and unbounded cases (polyadic or joining/splitting/full-coordination, respectively), these cannot be used to encode arity into coordination unless it could be encoded otherwise.

The interesting results here are the separation results that ensure no new encodings or expressiveness is going. The proof technique is clearly illustrated by the following result for the joining setting.

Theorem 5.1. *There exists no valid encoding from $\mathcal{L}_{A,P,D,NO,B}$ into $\mathcal{L}_{A,M,D,NO,J}$.*

Proof. The proof is by contradiction, assume there exists a valid encoding $[\![\cdot]\!]$. Consider the $\mathcal{L}_{A,P,D,NO,B}$ processes $P = [\langle a, b\rangle]\triangleleft$ and $Q = [(x, y)] \triangleright \surd$. Clearly it holds that $P \mid Q \longmapsto \surd$ and so $[\![P \mid Q]\!] \longmapsto$ and $[\![P \mid Q]\!] \Downarrow$ by validity of the encoding. Now consider the reduction $[\![P \mid Q]\!] \longmapsto$.

The reduction must be of the form $[\langle m_1\rangle] \triangleleft \mid \ldots \mid [\langle m_i\rangle] \triangleleft \mid [(z_1) \mid \ldots \mid (z_i)] \triangleright T'$ for some $\widetilde{m}$ and $\widetilde{z}$ and i and T'. Now consider the process whose encoding produces $[(z_1) \mid \ldots \mid (z_i)] \triangleright T'$, assume Q although the results do not rely on this assumption. If any $[\langle m_j\rangle]\triangleleft$ are also from the encoding of Q then it follows that the encoding of i instances of Q in parallel will reduce, i.e. $[\![Q \mid \ldots \mid Q]\!] \longmapsto$, while $Q \mid \ldots \mid Q \not\longmapsto$. Now consider two fresh processes S and T such that $S \mid T \longmapsto$ with some arity that is not 2 and $S \not\longmapsto$ and $T \not\longmapsto$. It follows that $[\![S \mid T]\!] \longmapsto$ (and $[\![S]\!] \not\longmapsto$ and $[\![T]\!] \not\longmapsto$) and $[\![S \mid T]\!]$ must include at least one $\langle n\rangle$ to do so. This $\langle n\rangle$ must arise from either $[\![S]\!]$ or $[\![T]\!]$, and conclude by showing that the encoding of i instances of either S or T in parallel with Q reduces, while the un-encoded processes do not. □

The splitting result is very similar with only minor adaptations to the proof.

Theorem 5.2. *There exists no valid encoding from $\mathcal{L}_{A,P,D,NO,B}$ into $\mathcal{L}_{A,M,D,NO,S}$.*

Proof. A straightforward adaptation of Theorem 5.1. □

Corollary 5.3. *If there exists no valid encoding from $\mathcal{L}_{\alpha_1,P,\gamma_1,\delta_1,B}$ into $\mathcal{L}_{\alpha_2,M,\gamma_2,\delta_2,B}$, then there exists no valid encoding from $\mathcal{L}_{\alpha_1,P,\gamma_1,\delta_1,\epsilon}$ into $\mathcal{L}_{\alpha_2,M,\gamma_2,\delta_2,\epsilon}$.*

Proof. The techniques in Theorems 5.1 & 5.2 apply to all joining and splitting languages, respectively. Name-matching requires only a small change of $Q = [(x, y)] \triangleright$ **if** $a = x$ **then** $\surd$ to then ensure binding occurs and not only name-matching; this then proved via contradiction of name invariance and success sensitiveness like in Theorem 4.9. The techniques in Theorem 6.3 more elegantly show that channel-based communication is insufficient than adding them here. Joining and splitting proofs can be combined for the full-coordination languages. □

The other main results are to show that existing encodings between binary languages can be reproduced in other forms of coordination. This turns out to be a straightforward adaptation of the usual techniques.

Consider the usual encoding of $\mathcal{L}_{S,P,D,NO,B}$ into $\mathcal{L}_{S,M,C,NO,B}$:

$$\begin{aligned} [\![\, [(\widetilde{a})] \lhd P \,]\!] &\stackrel{\text{def}}{=} (\nu c)[\overline{n}\langle c\rangle] \lhd [\overline{c}\langle a_1\rangle] \lhd \ldots \lhd [\overline{c}\langle a_n\rangle] \lhd [\![\, P \,]\!] \\ [\![\, [(\widetilde{x})] \rhd Q \,]\!] &\stackrel{\text{def}}{=} [n(z)] \rhd [z(x_1)] \rhd \ldots \rhd [z(x_n)] \rhd [\![\, Q \,]\!] \end{aligned}$$

where c is not in the free names of $[\langle\widetilde{a}\rangle] \lhd P$, and z is not in the free names of $[(\widetilde{x})] \rhd Q$ or $\{\widetilde{x}\}$. Also n is derived from $\widetilde{a}$ since $\widetilde{a} = a_1, \ldots, a_n$ (and similarly for $\widetilde{x}$). Thus when an output and input agree upon their arity n then they interact with the output sending a fresh name c used for sending the n names to be communicated.

This can be adapted in the obvious manner, shown below for the encoding of $\mathcal{L}_{S,P,D,NO,J}$ into $\mathcal{L}_{S,M,C,NO,J}$.

$$\begin{aligned} [\![\, [(\widetilde{a})] \lhd P \,]\!] &\stackrel{\text{def}}{=} (\nu c)[\overline{n}\langle c\rangle] \lhd [\overline{c}\langle a_1\rangle] \lhd \ldots \lhd [\overline{c}\langle a_n\rangle] \lhd [\![\, P \,]\!] \\ [\![\, [(\widetilde{x1}) \mid \ldots \mid (\widetilde{xk})] \rhd Q \,]\!] &\stackrel{\text{def}}{=} [n1(z_1) \mid \ldots \mid nk(z_k)] \rhd [z_1(x1_1)] \rhd \ldots \rhd [z_1(x1_{n1})] \rhd \\ &\qquad \ldots \rhd [z_k(xk_1)] \rhd \ldots \rhd [z_k(xk_{nk})] \rhd [\![\, Q \,]\!] \end{aligned}$$

where the restrictions on z are here extended to distinct $z_1, \ldots, z_k$ for each input in the join.

Theorem 5.4. *There is a valid encoding from $\mathcal{L}_{S,P,D,NO,J}$ into $\mathcal{L}_{S,M,C,NO,J}$.*

Proof. The proof technique is identical to Theorem 4.4, the only changes are straighforward adaptations of Lemma 4.1 and Lemma 4.2. □

This illustrates the key ideas for the following general result, that requires only straightforward adaptations of the proofs in the obvious manner. It is worth noting that all such results rely on the use of a channel-name, or an equivalent pattern match of some form to detect compatible arity and then ensure the right processes communicate. This is clearly available when adding channel-based communication, or when exploiting intensionality.

Theorem 5.5. *If there exists a valid encoding from $\mathcal{L}_{\alpha,P,\gamma,\delta,B}$ into $\mathcal{L}_{\alpha,M,\gamma,\delta,B}$ then there exists a valid encoding from $\mathcal{L}_{\alpha,P,\gamma,\delta,\epsilon}$ into $\mathcal{L}_{\alpha,M,\gamma,\delta,\epsilon}$.*

This confirms that encodings in the binary setting still exist in different coordination settings. It follows that no expressiveness differences between languages are lost by shifting to a different coordination form, and existing results can be transferred.

To conclude, any form of non-binary coordination does not allow for encoding polyadicity in a monadic language unless it could already be encoded by some other means.

6. Coordination and Communication Medium

This section considers the relation between coordination and communication medium. In general coordination is unable to encode communication medium unless it could otherwise be encoded by other features.

The base result for joining is illustrated in the following theorem, generalised in the corollary that follows.

Theorem 6.1. *There exists no valid encoding from $\mathcal{L}_{A,M,C,NO,B}$ into $\mathcal{L}_{A,M,D,NO,J}$.*

Proof. The proof is by contradiction, assume there exists a valid encoding $[\![\cdot]\!]$. Consider the $\mathcal{L}_{A,M,C,NO,B}$ processes $P = [\overline{a}\langle b\rangle]\triangleleft$ and $Q = [a(x)] \triangleright \surd$. Clearly it holds that $P \mid Q \longmapsto \surd$ and so $[\![P \mid Q]\!] \longmapsto$ and $[\![P \mid Q]\!] \Downarrow$ by validity of the encoding. Now consider the reduction $[\![P \mid Q]\!] \longmapsto$.

The reduction must be of the form $[\langle m_1\rangle] \triangleleft \mid \ldots \mid [\langle m_i\rangle] \triangleleft \mid [(z_1) \mid \ldots \mid (z_i)] \triangleright T'$ for some $\widetilde{m}$ and $\widetilde{z}$ and i and T'. Now consider the process whose encoding produces $[(z_1) \mid \ldots \mid (z_i)] \triangleright T'$, assume Q although the results do not rely on this assumption. If any $[\langle m_j\rangle]\triangleleft$ are also from the encoding of Q then it follows that the encoding of i instances of Q in parallel will reduce, i.e. $[\![Q \mid \ldots \mid Q]\!] \longmapsto$, while $Q \mid \ldots \mid Q \not\longmapsto$. Now consider two fresh processes $S = [\overline{c}\langle d\rangle]\triangleleft$ and $T = [c(z)] \triangleright \mathbf{0}$. Since $S \mid T \longmapsto$ it follows that $[\![S \mid T]\!] \longmapsto$ and must include at least one $\langle n\rangle$ to do so. This $\langle n\rangle$ must arise from either $[\![S]\!]$ or $[\![T]\!]$, and conclude by showing that the encoding of i instances of either S or T in parallel with Q reduces, while the un-encoded processes do not. □

Corollary 6.2. *If there exists no valid encoding from $\mathcal{L}_{\alpha_1,\beta_1,C,\delta_1,B}$ into $\mathcal{L}_{\alpha_2,\beta_2,D,\delta_2,B}$, then there exists no valid encoding from $\mathcal{L}_{\alpha_1,\beta_1,C,\delta_1,B}$ into $\mathcal{L}_{\alpha_2,\beta_2,D,\delta_2,J}$.*

Proof. The technique in Theorem 6.1 applies to all monadic languages (the addition of name-matching can be proved using the techniques as in Theorem 4.8). For the polyadic no-matching setting the result above holds by observing that the arity must remain fixed for an encoding, i.e. $[\![[\overline{a}\langle b_1, \ldots, b_i\rangle]\triangleleft]\!]$ is encoded to inputs/outputs all of some arity j. If the arity is not uniform then the encoding fails either operational correspondence (i.e. $[\![[a(x)] \triangleright \mathbf{0} \mid [\overline{a}\langle b_1, b_2\rangle]\triangleleft]\!] \longmapsto$) or divergence reflection as in sub-case (2) of Theorem 7.1 except here with arity instead of number of names. □

The base splitting result is similar to prior results.

Theorem 6.3. *There exists no valid encoding from $\mathcal{L}_{A,M,C,NO,B}$ into $\mathcal{L}_{A,M,D,NO,S}$.*

Proof. The proof is by contradiction and very similar to that of Theorem 5.2, the main differences are to consider the $\mathcal{L}_{A,M,C,NO,B}$ processes $P = [\overline{a}\langle b\rangle]\triangleleft$ and $Q = [a(x)] \triangleright \surd$, and then $S = [\overline{c}\langle d\rangle]\triangleleft$ and $T = [c(z)] \triangleright \mathbf{0}$. □

Corollary 6.4. *If there exists no valid encoding from $\mathcal{L}_{\alpha_1,\beta_1,C,\delta_1,B}$ into $\mathcal{L}_{\alpha_2,\beta_2,D,\delta_2,B}$, then there exists no valid encoding from $\mathcal{L}_{\alpha_1,\beta_1,C,\delta_1,\epsilon}$ into $\mathcal{L}_{\alpha_2,\beta_2,D,\delta_2,\epsilon}$.*

Proof. The results for joining languages are already by Corollary 6.2. The technique in Theorem 6.3 applies to all splitting monadic languages (the addition of name-matching can be proved using the techniques as in Theorem 4.8). For the polyadic no-matching setting the result above holds by observing that the arity must remain fixed for an encoding, i.e. $[\![\, [\overline{a}\langle b_1, \ldots, b_i\rangle]\triangleleft \,]\!]$ is encode to inputs/outputs all of some arity j. If the arity is not uniform then the encoding fails either operational correspondence (i.e. $[\![\, [a(x)]\triangleright \mathbf{0} \mid [\overline{a}\langle b_1, b_2\rangle]\triangleleft \,]\!] \longmapsto$) or divergence reflection as in case (2) of Theorem 7.2 except here with arity instead of number of names. The joining and splitting proofs can be combined for the full-coordination languages. □

Thus any form of non-binary coordination does not allow for encoding channels in a dataspace-based language unless it could already be encoded by some other means.

The positive encoding results are the typical adaptations of the positive encoding results in the binary setting. For example, consider the usual encoding from $\mathcal{L}_{S,P,C,NM,B}$ into $\mathcal{L}_{S,P,D,NM,B}$:

$$\begin{array}{rcl} [\![\, [\overline{a}\langle\widetilde{c}\rangle] \triangleleft P \,]\!] & \stackrel{\text{def}}{=} & [\langle a, \widetilde{c}\rangle] \triangleleft [\![\, P \,]\!] \\ [\![\, [a(\widetilde{x})] \triangleright Q \,]\!] & \stackrel{\text{def}}{=} & [(\ulcorner a \urcorner, \widetilde{x})] \triangleright [\![\, Q \,]\!] \,. \end{array}$$

The channel-name is simply moved to the first position in the polyadic input as a name-match. The adaptation of the encoding for $\mathcal{L}_{S,P,C,NM,J}$ into $\mathcal{L}_{S,P,D,NM,J}$ is the obvious one as below.

$$\begin{array}{rcl} [\![\, [\overline{a}\langle\widetilde{c}\rangle] \triangleleft P \,]\!] & \stackrel{\text{def}}{=} & [\langle a, \widetilde{c}\rangle] \triangleleft [\![\, P \,]\!] \\ [\![\, [a_1(\widetilde{x1}) \mid \ldots a_k(\widetilde{xk})] \triangleright Q \,]\!] & \stackrel{\text{def}}{=} & [(\ulcorner a_1 \urcorner, \widetilde{x1}) \mid \ldots \mid (\ulcorner a_k \urcorner, \widetilde{xk})] \triangleright [\![\, Q \,]\!] \end{array}$$

Here each input's channel is moved to the first position of it's polyadic input as a name-match.

Theorem 6.5. *There is a valid encoding from $\mathcal{L}_{S,P,C,NM,J}$ into $\mathcal{L}_{S,P,D,NM,J}$.*

Proof. The proof technique is identical to Theorem 4.4, the only changes are straighforward adaptations of Lemma 4.1 and Lemma 4.2. □

This illustrates the key ideas for the following general result, that requires only straightforward adaptations of the proofs in the obvious manner. Again all such results rely upon the use of pattern-matching, either via name-matching or intensionality, to represent the channel.

Theorem 6.6. *If there exists a valid encoding from $\mathcal{L}_{\alpha,\beta,C,\delta,B}$ into $\mathcal{L}_{\alpha,\beta,D,\delta,B}$ then there exists a valid encoding from $\mathcal{L}_{\alpha,\beta,C,\delta,\epsilon}$ into $\mathcal{L}_{\alpha,\beta,D,\delta,\epsilon}$.*

This confirms that encodings of channel-based communication into dataspace-based communication in the binary setting still exist in different coordination settings. It follows that no expressiveness differences between languages are lost by shifting to a different coordination form, and existing results can be transferred.

To conclude, any form of non-binary coordination does not allow for encoding channel-based communication into a dataspace-based language unless it could already be encoded by some other means.

7. Coordination and Pattern-Matching

This section considers the relations between coordination and pattern-matching. The great expressive power of name matching [24] and intensionality [19] prove impossible to encode with joining. This section formalises that despite this no greater form of coordination can be encoded into a lesser form by pattern matching.

The first result is to prove that intensionality cannot be encoded by coordination. Recall that since intensionality alone can encode all other features aside from coordination, it is sufficient to consider $\mathcal{L}_{A,M,D,I,B}$.

Theorem 7.1. *There exists no valid encoding from $\mathcal{L}_{A,M,D,I,B}$ into $\mathcal{L}_{-,-,-,\delta,J}$ where $\delta \neq I$.*

Proof. The proof is by contradiction. Assume there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{A,M,D,I,B}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,J}$ for some α and β and γ and δ where $\delta \neq I$. Consider the encoding of the processes $S_0 = [(x)] \triangleright [\langle m \rangle]\triangleleft$ and $S_1 = [\langle a \rangle]\triangleleft$. Clearly $[\![S_0 \mid S_1]\!] \longmapsto$ since $S_0 \mid S_1 \longmapsto$. There exists a reduction $[\![S_0 \mid S_1]\!] \longmapsto$ that must be between a join and some outputs that have combined maximal arity k. (The combined arity is the sum of the arities of all the input-patterns of the join involved, e.g. $[(a, b) \mid (c)] \triangleright \mathbf{0}$ has combined arity 3.)

Now define the following processes $S_2 \stackrel{\text{def}}{=} [\langle a_1 \bullet \ldots \bullet a_{2k+1} \rangle]\triangleleft$ and $S_3 \stackrel{\text{def}}{=} [(\ulcorner a_1 \urcorner \bullet \ldots \bullet \ulcorner a_{2k+1} \urcorner)] \triangleright [\langle m \rangle]\triangleleft$ where S_2 outputs $2k+1$ distinct names in a single term, and S_3 matches all of these names in a single intensional pattern.

Since $S_2 \mid S_0 \longmapsto$ it must be that $[\![S_2 \mid S_0]\!] \longmapsto$ for the encoding to be valid. Now consider the maximal combined arity of the reduction $[\![S_2 \mid S_0]\!] \longmapsto$.

- If the arity is k consider the reduction $[\![S_2 \mid S_3]\!] \longmapsto$ with the combined maximal arity j which must exist since $S_2 \mid S_3 \longmapsto$. Now consider the relationship of j and k.

    1. If $j = k$ then the upper bound on the number of names that are matched in the reduction is $2k$ (when each name is matched via a distinct channel). Since not all $2k+1$ tuples of names from $\varphi_{[\![\,]\!]}(a_i)$ can be matched in the reduction then there must be at least one tuple $\varphi_{[\![\,]\!]}(a_i)$ for $i \in \{1, \ldots, 2k+1\}$ that is not being matched in the interaction $[\![S_2 \mid S_3]\!] \longmapsto$. Now construct S_4 that differs from S_3 only by swapping one such name a_i with m: $S_4 \stackrel{\text{def}}{=} [(\ulcorner a_1 \urcorner \bullet \ldots \ulcorner a_{i-1} \urcorner \bullet \ulcorner m \urcorner \bullet \ulcorner a_{i+1} \urcorner \ldots \ulcorner a_{k+2} \urcorner)] \triangleright [\langle a_i \rangle]\triangleleft$. Now consider the context $C^{\mathcal{N}}_{|}([\![\, S_2 \,]\!], [\![\, \cdot \,]\!]) = [\![\, S_2 \mid \cdot \,]\!]$ where $N = \{\widetilde{a} \cup m\}$. Clearly neither $C^{\mathcal{N}}_{|}([\![\, S_2 \,]\!], [\![\, \mathbf{0} \,]\!]) \longmapsto$ nor $C^{\mathcal{N}}_{|}([\![\, S_2 \,]\!], [\![\, S_4 \,]\!]) \longmapsto$ as this would contradict Proposition 3.1. However, since S_3 and S_4 differ only by the position of one name whose tuple $\varphi_{[\![\,]\!]}(\cdot)$ does not appear in the reduction $[\![\, S_2 \mid S_3 \,]\!] \longmapsto$, it follows that the reason $C^{\mathcal{N}}_{|}([\![\, S_2 \,]\!], [\![\, S_4 \,]\!]) \not\longmapsto$ must be due to a structural congruence difference between $C^{\mathcal{N}}_{|}([\![\, S_2 \,]\!], [\![\, S_3 \,]\!])$ and $C^{\mathcal{N}}_{|}([\![\, S_2 \,]\!], [\![\, S_4 \,]\!])$. Further, by compositionality of the encoding the difference can only be between $[\![\, S_3 \,]\!]$ and $[\![\, S_4 \,]\!]$. Since Proposition 3.1 ensures that $[\![\, S_3 \,]\!] \not\longmapsto$ and $[\![\, S_4 \,]\!] \not\longmapsto$, the only possibility is a structural difference between $[\![\, S_3 \,]\!]$ and $[\![\, S_4 \,]\!]$. Now exploiting $\sigma = \{m/a_i, a_i/m\}$ such that $\sigma S_4 = S_3$ yields contradiction.

2. If $j \neq k$ then obtain that $[\![S_2]\!]$ must be able to interact with both combined arity k and combined arity j. That is, $[\![S_2 \mid \cdot]\!] = C^N_{|}([\![S_2]\!], [\![\cdot]\!])$ where $N = \{\widetilde{a} \cup m\}$ and that $C^N_{|}([\![S_2]\!], [\![S_0]\!])$ reduces with combined arity k and $C^N_{|}([\![S_2]\!], [\![S_3]\!])$ reduces with combined arity j. Now it is straightforward, if tedious, to show that since $S_0 \mid S_3 \not\longmapsto$ that $C^N_{|}([\![S_2]\!], [\![S_0 \mid S_3]\!])$ can perform the same initial reductions as either $C^N_{|}([\![S_2]\!], [\![S_0 \mid \mathbf{0}]\!])$ or $C^N_{|}([\![S_2]\!], [\![\mathbf{0} \mid S_3]\!])$ by exploiting operational correspondence and Proposition 3.1.
   Thus, it can be shown that $C^N_{|}([\![S_2]\!], [\![S_0 \mid S_3]\!])$ can perform both the k combined arity reduction of $[\![S_2 \mid S_0]\!] \longmapsto$ and the j combined arity reduction of $[\![S_2 \mid S_3]\!] \longmapsto$. Now by exploiting the structural congruence rules it follows that neither of these initial reductions can prevent the other occurring. Thus, $C^N_{|}([\![S_2]\!], [\![S_0 \mid S_3]\!])$ must be able to do both of these initial reductions in any order.
   Now consider the process R that has performed both of these initial reductions. By operational correspondence it must be that $R \not\Longrightarrow\simeq [\![\, [\langle m \rangle] \lhd \mid [\langle m \rangle] \lhd \,]\!]$ since $S_2 \mid S_0 \mid S_3 \not\Longrightarrow [\langle m \rangle] \lhd \mid [\langle m \rangle] \lhd$. Therefore, R must be able to roll-back the initial step with combined arity j; i.e reduce to a state that is equivalent to the reduction not occurring. (Or the initial step with arity k, but either one is sufficient as by operational correspondence $R \Longrightarrow\simeq [\![\, [\langle m \rangle] \lhd \mid S_3 \,]\!]$.)
   Now consider how many names are being matched in the initial reduction with combined arity j. If $j < k$ the technique of differing on one name used in the case of $j = k$ can be used to show that this would introduce divergence on the potential roll-back and thus contradict a valid encoding. Therefore it must be that $j > k$. Finally, by exploiting name invariance and substitutions like $\{(b_1 \bullet \ldots \bullet b_j)/a_1\}$ applied to S_2 and S_3 it follows that either $j > k + j$ or both S_2 and S_3 must have infinitely many initial reductions which yields divergence.

- If the combined arity is not k then proceed like the second case above. □

Theorem 7.2. *There exists no valid encoding from $\mathcal{L}_{A,M,D,I,B}$ into $\mathcal{L}_{-,-,-,\delta,S}$ where $\delta \neq I$.*

Proof. The same technique as in Theorem 7.1 can be applied for splitting. □

Corollary 7.3. *If there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,I,B}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,B}$, then there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,I,\epsilon}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,\epsilon}$.*

Proof. The joining case is covered by Theorem 7.1 and the splitting by Theorem 7.2. The full-coordination case is a by a straightforward adaptation of either of these since neither rely on particular aspects of joining or splitting. □

It follows that any form of coordination cannot represent intensionality in a language that does not have intensionality already (including name-matching or no-matching languages). The next results show that name matching is insufficient to encode coordination.

Theorem 7.4. *There exists no valid encoding from $\mathcal{L}_{A,M,D,NM,B}$ into $\mathcal{L}_{\alpha,\beta,\gamma,NO,J}$.*

Proof. The proof is by contradiction, assume there exists a valid encoding $[\![\cdot]\!]$. Consider the $\mathcal{L}_{A,M,D,NM,B}$ processes $P = [\langle a \rangle]\triangleleft$ and $Q = [(\ulcorner a \urcorner)] \triangleright ([\langle b \rangle] \triangleleft \mid \surd)$. Clearly it holds that $P \mid Q \longmapsto$ and $P \mid Q \Downarrow$ and so $[\![P \mid Q]\!] \longmapsto$ and $[\![P \mid Q]\!] \Downarrow$ by validity of the encoding. Now consider γ.

- If $\gamma = D$ then consider the substitution $\sigma = \{a/b, b/a\}$, it is clear that $P \mid \sigma Q \not\longmapsto$ and so $[\![P \mid \sigma Q]\!] \not\longmapsto$, however the only possibility that this holds is when $[\![\sigma Q]\!]$ is blocked from interacting. It is then straightforward if tedious to show that any such blocking of reduction would either imply $[\![\sigma(P \mid Q)]\!] \not\longmapsto$ or $\sigma(P \mid Q) \not\longmapsto$ and thus contradict the validity of the encoding.

- Otherwise it must be that $\gamma = C$. Now consider the reduction $[\![P \mid Q]\!] \longmapsto$ that must be of the form $[\overline{c_1}\langle\widetilde{m_1}\rangle] \triangleleft \mid \ldots \mid [\overline{c_i}\langle\widetilde{m_i}\rangle] \triangleleft \mid [c_1(\widetilde{z_1}) \mid \ldots \mid c_i(\widetilde{z_i})] \triangleright T_1$ for some $\widetilde{c}$ and $\widetilde{m}$ and $\widetilde{z}$ and i and T_1. Again consider the substitution $\sigma = \{a/b, b/a\}$, it is clear that $\sigma P \mid Q \not\longmapsto$ and so $[\![\sigma P \mid Q]\!] \not\longmapsto$. The only way this can occur without contradicting the validity of the encoding (as in the previous case) is when there is at least one c_k in the domain of some σ' where $\sigma'(c_k) \neq c_k$ and $[\![\sigma P]\!] \simeq \sigma' [\![P]\!]$ by definition of the encoding.

  Now consider the process $S = [(x)] \triangleright S'$, clearly $P \mid S \longmapsto$ and so $[\![P \mid S]\!] \longmapsto$ as well. The reduction $[\![P \mid S]\!] \longmapsto$ must be from the form $[\overline{d_1}\langle\widetilde{n_1}\rangle]\triangleleft \mid \ldots \mid [\overline{d_j}\langle\widetilde{n_j}\rangle]\triangleleft \mid [d_1(\widetilde{w_1}) \mid \ldots \mid d_j(\widetilde{w_j})] \triangleright T_2$ for some $\widetilde{d}$ and $\widetilde{n}$ and $\widetilde{w}$ and j and T_2.

  Now if $i = j$ it follows that for each $k \in \{1 \ldots i\}$ then $c_k = d_k$. However, this contradicts the validity of the encoding since there is some c_k in the domain of σ' such that $\sigma'(c_k) \neq c_k$ and $\sigma P \mid S \longmapsto$ while $[\![\sigma P \mid S]\!] \not\longmapsto$.

  Otherwise it must be that $i > j$ (otherwise if $i < j$ then $[\![P \mid S]\!] \not\longmapsto$) and that $c_k \in \{c_{j+1}, \ldots, c_i\}$. Now consider when $S' = \textbf{if } x = a \textbf{ then } \Omega$, clearly $P \mid S \longmapsto\equiv \Omega$ and $\sigma P \mid S \longmapsto\equiv \mathbf{0}$ and so $[\![P \mid S]\!]$ diverges and $[\![\sigma P \mid S]\!] \Longmapsto\simeq \mathbf{0}$. Now it can be shown that $P \mid \sigma P \mid S \mid Q \longmapsto\longmapsto\equiv \surd$ while $[\![P \mid \sigma P \mid S \mid Q]\!] \Downarrow$ and diverges since $[\![\sigma P]\!]$ can satisfy the first j input patterns of $[\![Q]\!]$ and $[\![\sigma P]\!]$ the remaining $i - j$, leaving the first j input patterns of $[\![P]\!]$ to interact with $[\![S]\!]$ and yield divergence. The only other possibility is that $[\![P \mid \sigma P \mid S \mid Q]\!] \not\Downarrow$. However, this requires that T_1 check some binding name in $\widetilde{z}$ for equality with a before yielding success (i.e. **if** $z_1 = a$ **then** $\surd$). This can in turn be shown to contradict the validity of the encoding by adding another instance of P. □

Theorem 7.5. *There exists no valid encoding from $\mathcal{L}_{A,M,D,NM,B}$ into $\mathcal{L}_{\alpha,\beta,\gamma,NO,S}$.*

Proof. The same technique as in Theorem 7.4 can be applied here. □

Corollary 7.6. *If there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,NM,B}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,B}$, then there exists no valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,NM,\epsilon}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta,\epsilon}$.*

Proof. The joining case is covered by Theorem 7.4 and the splitting by Theorem 7.5. The full-coordination case is a by a straightforward adaptation of either of these since neither rely on particular aspects of joining or splitting. □

Thus coordination does not allow for encoding name-matching into a no-matching language unless it could already be encoded by some other means.

For the positive results that encodings remain it is straightforward to adapt the existing encodings in the same manner as for Corollary 4.7, and Theorems 5.5, & 6.6.

Theorem 7.7. *If there exists a valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,\delta_1,B}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta_2,B}$ where $\delta_1 \leq \delta_2$ then there exists a valid encoding from $\mathcal{L}_{\alpha,\beta,\gamma,\delta_1,\epsilon}$ into $\mathcal{L}_{\alpha,\beta,\gamma,\delta_2,\epsilon}$.*

Proof. The same techniques as Corollary 4.7, and Theorems 5.5, & 6.6 can be applied. □

Finally, the positive results that preserve encodings when changing the coordination feature can be combined into a single general result.

Corollary 7.8. *If there exists a valid encoding from $\mathcal{L}_{\alpha_1,\beta_1,\gamma_1,\delta_1,B}$ into $\mathcal{L}_{\alpha_2,\beta_2,\gamma_2,\delta_2,B}$ where $\gamma_1 \leq \gamma_2$ then there exists a valid encoding from $\mathcal{L}_{\alpha_1,\beta_1,\gamma_1,\delta_1,\epsilon}$ into $\mathcal{L}_{\alpha_2,\beta_2,\gamma_2,\delta_2,\epsilon}$.*

Proof. By combining Corollary 4.7, and Theorems 5.5, 6.6, & 7.7. □

8. Coordination and Other Features

This section considers the expressive power gained by coordination. It turns out that coordination adds expressive power that cannot be represented by binary languages regardless of other features.

The expressive power gained by joining or splitting can be captured by the concept of the *coordination degree* of a language $\mathcal{L}$, denoted $\textsc{Cd}(\mathcal{L})$, as the least upper bound on the number of processes that must coordinate to yield a reduction in $\mathcal{L}$. For example, all the binary languages $\mathcal{L}_{-,-,-,-,B}$ have coordination degree 2 since their reduction axiom is only defined for two processes. By contrast, the coordination degree of the non-binary languages is ∞ since there is no bound on the number of inputs that can be part of a join, or outputs that can be part of a split.

Theorem 8.1. *If $\textsc{Cd}(\mathcal{L}_1) > \textsc{Cd}(\mathcal{L}_2)$ then there exists no valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_1$ into $\mathcal{L}_2$.*

Proof. By contradiction, assume there is a valid encoding $[\![\cdot]\!]$. Pick i processes S_1 to S_i where $i = \textsc{Cd}(\mathcal{L}_2) + 1$ such that all these processes must coordinate to yield a reduction and yield success. That is, $S_1 \mid \ldots \mid S_i \longmapsto \surd$ but not if any S_j (for $1 \leq j \leq i$) is replaced by the null process $\mathbf{0}$. By validity of the encoding it must be that $[\![S_1 \mid \ldots \mid S_i]\!] \longmapsto$ and $[\![S_1 \mid \ldots \mid S_i]\!] \Downarrow$.

By compositionality of the encoding $[\![S_1 \mid \ldots \mid S_i]\!] = C_S$ where C_S must be of the form $C_{\mid}^{N}([\![S_1]\!], C_{\mid}^{N}(\ldots, C_{\mid}^{N}([\![S_{i-1}]\!], [\![S_i]\!])\ldots))$. Now consider the reduction $[\![S_1 \mid \ldots \mid S_i]\!] \longmapsto$ that can be at most between $i-1$ processes by the coordination degree of $\mathcal{L}_2$. If the reduction does *not* involve some process $[\![S_j]\!]$ then it follows that $[\![S_1 \mid \ldots \mid S_{j-1} \mid \mathbf{0} \mid S_{j+1} \mid \ldots \mid S_i]\!] \longmapsto$ (by replacing the $[\![S_j]\!]$ in the context C_S with $[\![\mathbf{0}]\!]$). By construction of $S_1 \mid \ldots \mid S_i$ and $\textsc{Cd}(\mathcal{L}_2) < i$ there must exist some such S_j. However, this contradicts the validity of the encoding since $S_1 \mid \ldots \mid S_{j-1} \mid \mathbf{0} \mid S_{j+1} \mid \ldots \mid S_i \not\longmapsto$. The only other possibility is if $[\![S_j]\!]$ blocks the

reduction by blocking some $[\![S_k]\!]$. This can only occur when $[\![S_k]\!]$ is either underneath an interaction primitive (e.g. $[\overline{s}\langle\widetilde{t}\rangle] \triangleleft [\![S_k]\!]$) or inside a conditional (e.g. **if** $s = t$ **then** $[\![S_k]\!]$). Both require that $[\![S_k]\!]$ not be top level in C_S, which can be proven contradictory by $i - 1$ applications of Proposition 3.2. □

Corollary 8.2. *There exists no valid encoding from $\mathcal{L}_{-,-,-,-,\epsilon}$ into $\mathcal{L}_{-,-,-,-,B}$ where $\epsilon \neq B$.*

The following result concludes the relations for increases in the coordination feature since all the non-binary languages have infinite coordination degree.

Theorem 8.3. *There exists no valid encoding from $\mathcal{L}_{-,-,-,-,F}$ into any language $\mathcal{L}_{-,-,-,-,\epsilon}$ where $\epsilon \neq F$.*

Proof. By contradiction, assume there is a valid encoding and consider the $\mathcal{L}_{A,M,D,NO,F}$ processes $P = [\langle a_1\rangle \mid \ldots \mid \langle a_i\rangle]\triangleleft$ and $Q = [(x_1) \mid \ldots \mid (x_i)] \triangleright Q'$. Since $P \mid Q \longmapsto$ it follows by validity of the encoding that $[\![P \mid Q]\!] \longmapsto$. Now consider ϵ.

- If $\epsilon = J$ then the reduction $[\![P \mid Q]\!] \longmapsto$ must be of the form $R_1 \mid R_2 \longmapsto$ from some $R_1 = [s_1(\widetilde{p_1}) \mid \ldots \mid s_j(\widetilde{p_j})] \triangleright S$ and $\widetilde{s}$ and $\widetilde{p}$ and j and S and R_2. Observe that $R_1 \mid R_2$ cannot be a reduct of either $[\![P]\!]$ or $[\![Q]\!]$ since this would violate Proposition 3.1. Further, it can be shown that R_1 must arise from $[\![P]\!]$ since otherwise there would be some process T that does all except for some x_k inputs of Q as separate processes like $T = [(x_1)] \triangleright T'_1 \mid \ldots \mid [(x_{k-1})] \triangleright T'_{k-1} \mid [(x_{k+1})] \triangleright T'_{k+1} \mid \ldots \mid [(x_i)] \triangleright T'_i$ such that $[\![P \mid T]\!] \longmapsto$ while $P \mid T \not\longmapsto$ and this would contradict operational correspondence. However, the same technique can be used to show that R_1 must arise from $[\![Q]\!]$, which is contradictory unless both $[\![P]\!]$ and $[\![Q]\!]$ have such a top level join. If both $[\![P]\!]$ and $[\![Q]\!]$ have such a top level join then by considering the processes $P_1 = [\langle c\rangle]\triangleleft$ and $Q_1 = [(x_1)] \triangleright Q'$ and $Q' =$ **if** $x_1 = a_1$ **then** Ω **else if** $x_1 = c$ **then** $\surd$ and the substitution $\sigma = \{c/a_1, a_1/c\}$ it can be shown that $P \mid Q \mid \sigma(P_1 \mid Q_1)$ can diverge or report success but not both, however it can be shown that $[\![P \mid Q \mid \sigma(P_1 \mid Q_1)]\!]$ can both diverge and report success.

- If $\epsilon = S$ then proceed in the same manner as the $\epsilon = J$ case above.

- If $\epsilon = B$ then Corollary 8.2 is sufficient. □

In the other direction the result is ensured by Remark 2.1. Thus for any languages $\mathcal{L}_{\alpha,\beta,\gamma,\delta,\epsilon_1}$ and $\mathcal{L}_{\alpha,\beta,\gamma,\delta,\epsilon_2}$ where $\epsilon_1 \leq \epsilon_2$ then it holds that $\mathcal{L}_{\alpha,\beta,\gamma,\delta,\epsilon_2}$ is strictly more expressive than $\mathcal{L}_{\alpha,\beta,\gamma,\delta,\epsilon_1}$. That is, joining or splitting languages are strictly more expressive than binary languages, and full coordination languages are strictly more expressive than either joining or splitting languages.

Thus coordination turns out to be orthogonal to all other features, since from the prior sections coordination cannot encode any other feature, and here it is proven that other features cannot encode coordination.

9. Within Coordination

This section considers relations between different forms of coordination. It turns out that there are some encodings from joining languages into splitting languages and vice versa, however most joining and splitting languages cannot be encode one another.

A joining (alt. splitting) language that is no-matching can be encoded into a splitting (resp. joining) language that can represent channel-based communication. For example, consider the encoding from $\mathcal{L}_{S,M,C,NO,J}$ to $\mathcal{L}_{S,M,C,NO,S}$ that is the identity on all forms except the output and join as follows:

$$\begin{aligned} [\![\, [\overline{a}\langle b\rangle] \lhd P \,]\!] &\stackrel{\text{def}}{=} [a(c)] \rhd [\overline{c}\langle b\rangle] \lhd [\![\, P \,]\!] \\ [\![\, [a_1(x_1) \mid \ldots \mid a_i(x_i)] \rhd Q \,]\!] &\stackrel{\text{def}}{=} (\nu \widetilde{c})([\overline{a_1}\langle c_1\rangle \mid \ldots \mid \overline{a_i}\langle c_i\rangle] \lhd \\ &\qquad [c_1(x_1)] \rhd \ldots \rhd [c_i(x_i)] \rhd [\![\, Q \,]\!]) \end{aligned}$$

where c is not b or in the free names of P; and $\widetilde{c}$ does not intersect with $\widetilde{a}$ or $\widetilde{x}$ or the free names of Q. The key idea is that the direction of communication is reversed; outputs become inputs, and joins become splits, a fresh name c is transmitted to be used for then sending the original name b from the output to the encoded join. Thus the requirement that all inputs of a join interact at once is maintained by all the outputs of the split.

Lemma 9.1. *Given a $\mathcal{L}_{S,M,C,NO,J}$ join P and output Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by induction on the number of input patterns in P and then for each one by definition of the poly-match rule. □

Lemma 9.2. *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. The proof is trivial for all the primitives except input and output as they are translated homomorphically. The output is straightforward, the input is by induction on the number of inputs in the join. □

Lemma 9.3. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{S,M,C,NO,J}$ into $\mathcal{L}_{S,M,C,NO,S}$ preserves and reflects reductions.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 9.1, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and synchronisation rule. The size of k in both cases is $2+i$ where i is the number of input-patterns of the input involved in $P \longmapsto P'$. The inductive cases where the last rule used is a structural one then rely on Lemma 9.2. □

Theorem 9.4. *The encoding from $\mathcal{L}_{S,M,C,NO,J}$ into $\mathcal{L}_{S,M,C,NO,S}$ is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 9.3. Success sensitiveness can be proved as follows: $P \Downarrow$ means

that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 9.3 k times and Lemma 9.2 obtain that $[\![P]\!] \longmapsto^j [\![P']\!] \equiv [\![P'']\!] \mid \surd$ where j can be determined from the instantiations of Lemma 9.2, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. □

The same approach can be used to encode $\mathcal{L}_{S,M,C,NO,S}$ into $\mathcal{L}_{S,M,C,NO,J}$ (the identity on all forms except) with the split and input as follows:

$$[\![\, [\overline{a_1}\langle b_1\rangle \mid \ldots \mid \overline{a_i}\langle b_i\rangle] \lhd P \,]\!] \stackrel{\text{def}}{=} [a_1(c_1) \mid \ldots \mid a_i(c_i)] \rhd [\overline{c_1}\langle b_1\rangle] \lhd \ldots \lhd [\overline{c_i}\langle b_i\rangle] \lhd [\![\, P \,]\!]$$

$$[\![\, [a(x)] \rhd Q \,]\!] \stackrel{\text{def}}{=} (\nu c)[\overline{a}\langle c\rangle] \lhd [c(x)] \rhd [\![\, Q \,]\!]$$

$\widetilde{c}$ does not intersect $\widetilde{b}$ or the free names of P; and c is not a or in the free names of Q.

Theorem 9.5. *The encoding from $\mathcal{L}_{S,M,C,NO,S}$ into $\mathcal{L}_{S,M,C,NO,J}$ is valid.*

Proof. The proof is similar to Theorem 9.4. □

The same techniques can be applied to the polyadic variations of the above languages, and to the asynchronous variations as well.

Theorem 9.6. *The languages $\mathcal{L}_{-,\beta,C,NO,J}$ and $\mathcal{L}_{-,\beta,C,NO,S}$ can validly encode each other.*

Proof. The proof is similar to Theorem 9.4. □

However there are usually not encodings between joining and splitting languages. This can be illustrated by considering attempts to encode any sort of name-matching from either joining or splitting into the other.

Theorem 9.7. *There exists no valid encoding from $\mathcal{L}_{A,M,D,NM,J}$ into $\mathcal{L}_{-,-,-,-,S}$.*

Proof. The proof is by contradiction. Consider the processes $P = [(\ulcorner a \urcorner) \mid (\ulcorner b \urcorner)] \rhd P'$ and $Q_1 = [\langle a\rangle]\lhd$ and $Q_2 = [\langle b\rangle]\lhd$. Since $P \mid Q_1 \mid Q_2 \longmapsto$ by validity of the encoding $[\![\, P \mid Q_1 \mid Q_2 \,]\!] \longmapsto$ so consider this reduction. It must be between some $R_1 = [\overline{s_1}\langle\widetilde{t_1}\rangle \mid \ldots \mid \overline{s_i}\langle\widetilde{t_i}\rangle] \lhd R_1'$ and R_2 for some $\widetilde{s}$ and $\widetilde{t}$ and R_1' and R_2 such that $R_1 \mid R_2 \longmapsto$. Observe that $R_1 \mid R_2$ cannot be a reduct of $[\![\, P \mid Q_1 \mid \mathbf{0} \,]\!]$ or $[\![\, P \mid \mathbf{0} \mid Q_2 \,]\!]$ or $[\![\, \mathbf{0} \mid Q_1 \mid Q_2 \,]\!]$ since this would contradict Proposition 3.1. It can be shown that R_1 must arise from $[\![\, P \,]\!]$ since otherwise the reduction would not require all components of the encoded processes, and so replacing some Q_i with the null process in $[\![\, P \mid Q_1 \mid Q_2 \,]\!]$ would still reduce while the unencoded process would not.

Now consider the process $S = [(z)] \rhd S'$ such that $Q_1 \mid S \longmapsto$ and $[\![\, Q_1 \mid S \,]\!] \longmapsto$ by operational correspondence. Observe that $[\![\, Q_1 \,]\!]$ interacts with $[\![\, P \,]\!]$ via some $[s_j(\widetilde{p})] \rhd Q_1'$ (there may be many such, but assume one for simplicity since the following can be proved for all of them). Now consider the reduction $[\![\, Q_1 \mid S \,]\!] \longmapsto$:

- If it is via the same $[s_j(\widetilde{p})] \rhd Q_1'$ that interacts with $[\![\, P \,]\!]$ then there must be some $[\ldots \mid \overline{s_j}\langle\widetilde{t}\rangle \mid \ldots] \lhd T'$ in $[\![\, S \,]\!]$ such that $\textsc{Match}(\widetilde{t}, \widetilde{p})$ is defined. Observe that this must not rely on equality/matching of any names that depend upon a since otherwise the substitution $\sigma = \{c/a\}$ would prevent the reduction of $Q_1 \mid \sigma S$ yet $Q_1 \mid \sigma S \longmapsto$ and so this would contradict name invariance. However, since no name in $[s_j(\widetilde{p})] \rhd Q_1'$ depends upon a it follows that $[\![\, P \mid \sigma Q_1 \mid Q_2 \,]\!] \longmapsto$ which contradicts Proposition 3.1.

- Otherwise it must be that the reduction is via some different input or output in $[\![Q_1]\!]$. However, this can be shown to yield divergence or violate operational correspondence in a similar manner to case (2) of Theorem 7.2.

□

Theorem 9.8. *There exists no valid encoding from $\mathcal{L}_{A,M,D,NM,S}$ into $\mathcal{L}_{-,-,-,-,J}$.*

Proof. The proof is by contradiction in a similar manner to Theorem 9.7 by starting with the processes $P = [\langle a \rangle \mid \langle b \rangle]_{\triangleleft}$ and $Q_1 = [(\ulcorner a \urcorner)] \triangleright Q_1'$ and $Q_2 = [(\ulcorner b \urcorner)] \triangleright Q_2'$. □

These results show that once name-matching (or intensionality) is in play it is no longer possible for splitting or joining languages to encode one another.

Corollary 9.9. *There exists no valid encoding from $\mathcal{L}_{A,M,D,I,J}$ into $\mathcal{L}_{-,-,-,-,S}$.*

Corollary 9.10. *There exists no valid encoding from $\mathcal{L}_{A,M,D,I,S}$ into $\mathcal{L}_{-,-,-,-,J}$.*

In the other direction once channels names are no longer representable it becomes impossible to encode joining into splitting or vice versa. The following results suffice since adding channel-based communication or name-matches (here with a polyadic target language) is sufficient to support channel-based communication and thus encoding.

Theorem 9.11. *There exists no valid encoding from $\mathcal{L}_{A,M,D,NO,J}$ into $\mathcal{L}_{-,-,D,NO,S}$.*

Proof. The proof is by contradiction, consider the processes $P = [(x) \mid (y)] \triangleright P'$ and $Q_1 = [\langle a \rangle]_{\triangleleft}$ and $Q_2 = [\langle b \rangle]_{\triangleleft}$. Since $P \mid Q_1 \mid Q_2 \longmapsto$ it follows that $[\![P \mid Q_1 \mid Q_2]\!] \longmapsto$ by operational correspondence and validity of the encoding. Now consider the reduction $[\![P \mid Q_1 \mid Q_2]\!] \longmapsto$ that must be between some $R_1 = [\langle \widetilde{t_1} \rangle \mid \ldots \mid \langle \widetilde{t_i} \rangle] \triangleleft R_1'$ and R_2 for some $\widetilde{t}$ and R_1' and R_2. Clearly $R_1 \mid R_2$ cannot be a reduct of $[\![P \mid Q_1 \mid \mathbf{0}]\!]$ or $[\![P \mid \mathbf{0} \mid Q_2]\!]$ or $[\![\mathbf{0} \mid Q_1 \mid Q_2]\!]$ since this would contradict Proposition 3.1. Further it can be shown that R_1 must arise from $[\![P]\!]$ since otherwise the reduction would not require all components of the encoded processes.

It follows that after the initial interaction there must be some input $[(\widetilde{p})] \triangleright T$ for some $\widetilde{p}$ and T that arises from $[\![P]\!]$ since otherwise by taking $P' = \mathbf{if}\ x = a\ \mathbf{then}\ \Omega$ it can be that $[\![P]\!]$ must always diverge after interaction or never diverge, which contradicts divergence reflection since $P \mid Q_1 \mid Q_1$ diverges but $P \mid Q_2 \mid Q_2$ does not. Now consider when $P' = \mathbf{if}\ x = a\ \mathbf{then}\ \Omega\ \mathbf{else\ if}\ x = c\ \mathbf{then}\ \surd$ and the substitution $\sigma = \{c/a, a/c\}$ and $S = P \mid Q_1 \mid Q_2 \mid \sigma(P \mid Q_1 \mid Q_2)$. Observe that S can diverge or report success but cannot do both. However, it can be shown that $[\![P \mid Q_1 \mid Q_2 \mid \sigma(P \mid Q_1 \mid Q_2)]\!]$ can both diverge and report success. □

Theorem 9.12. *There exists no valid encoding from $\mathcal{L}_{A,M,D,NO,S}$ into $\mathcal{L}_{-,-,D,NO,J}$.*

Proof. The proof is similar to Theorem 9.11 by considering the processes $P = [\langle a \rangle \mid \langle b \rangle]_{\triangleleft}$ and $Q_1 = [(x)] \triangleright Q_1'$ and $Q_2 = [(y)] \triangleright Q_2'$. Since $P \mid Q_1 \mid Q_2 \longmapsto$ it follows that $[\![P \mid Q_1 \mid Q_2]\!] \longmapsto$ by operational correspondence and validity of the encoding. Now consider the reduction $[\![P \mid Q_1 \mid Q_2]\!] \longmapsto$ that must be between some $R_1 = [(\widetilde{p_1}) \mid \ldots \mid (\widetilde{p_i})] \triangleright R_1'$ and R_2 for some $\widetilde{p}$ and R_1' and R_2. Clearly $R_1 \mid R_2$ cannot be a reduct of $[\![P \mid Q_1 \mid \mathbf{0}]\!]$ or $[\![P \mid \mathbf{0} \mid Q_2]\!]$ or $[\![\mathbf{0} \mid Q_1 \mid Q_2]\!]$ since this would contradict

Proposition 3.1. Further it can be shown that R_1 must arise from $[\![P]\!]$ since otherwise the reduction would not require all components of the encoded processes.

It follows that after the initial interaction there must be some input $[(\widetilde{q})] \triangleright T$ for some $\widetilde{p}$ and T that arises from $[\![Q_1]\!]$ since otherwise taking $Q_1' = \textbf{if } x = a \textbf{ then } \Omega$ and the substitution $\sigma = \{c/a\}$ it can be shown that $[\![Q_1]\!]$ must always diverge after interaction or never diverge, which contradicts divergence reflection since $P \mid Q_1 \mid Q_2$ can diverge but $\sigma P \mid Q_1 \mid Q_2$ cannot. Now consider when $Q_1' = \textbf{if } x = a \textbf{ then } \Omega \textbf{ else if } x = c \textbf{ then } \surd$ and the substitution $\rho = \{c/a, a/c\}$ and $S = P \mid Q_1 \mid Q_2 \mid \sigma(P \mid Q_1 \mid Q_2)$. Observe that S can diverge or report success but cannot do both. However, it can be shown that $[\![P \mid Q_1 \mid Q_2 \mid \sigma(P \mid Q_1 \mid Q_2)]\!]$ can both diverge and report success since the input $[(\widetilde{q})] \triangleright T$ cannot distinguish between the partial reductions. □

Thus although there are some languages where a difference only of joining or splitting prove equally expressive, in general different forms of coordination usually indicate differences in expressive power.

10. Conclusions and Future Work

Languages with non-binary coordination have been considered before, although less often than binary languages. It turns out that increases in coordination degree correspond to increases in expressive power. For example, an intensional binary language cannot be encoded by a non-intensional joining language. However, encodings from lesser coordination languages into greater coordination languages are still dependent upon other features.

This formalises that languages like the Join Calculus, general rendezvous calculus, and m-calculus cannot be validly encoded into binary languages, regardless of other features. Although there exist encodings from (for example) Join Calculus into π-calculus [15] these do not meet the criteria for a *valid encoding* used here. The general approach used in such encodings is to encode joins by $[\![[m(x) \mid n(y)] \triangleright P]\!] = m(x).n(y).[\![P]\!]$, however this can easily fail operational correspondence, divergence reflection, or success sensitivity. For example consider $P_1 = [c_1(w) \mid c_2(x)] \triangleright \surd$ and $P_2 = [c_2(y) \mid c_1(z)] \triangleright \Omega$ and $Q = \overline{c_1}\langle a \rangle \mid \overline{c_2}\langle b \rangle$. Together $P_1 \mid P_2 \mid Q$ can either report success or diverge, but their encoding $[\![P_1 \mid P_2 \mid Q]\!]$ can deadlock. Even ordering the channel names to prevent this can be shown to fail under substitutions.

In general the coordination feature is unrelated to any of the other features. That is: none of synchronicity, arity, communication-medium, or pattern-matching can be encoded by coordination. Similarly, none of the other features can encode greater coordination features into lesser ones; full-coordination languages cannot be encoded into joining or splitting languages, and neither joining nor splitting languages can be encoded into binary languages.

Apart from these more general results, it turns out that joining and splitting languages can sometimes encode one another. In particular, a source language without name matching or intensionality can be encoded into a target language that can represent channel based communication. Indeed, this holds both from joining into splitting, and splitting into joining. This further reinforced prior results that pattern matching is the most significant feature for understanding expressiveness.

[1] Bengtson, J., Johansson, M., Parrow, J., Victor, B., 2011. Psi-calculi: a framework for mobile processes with nominal data and logic. Logical Methods in Computer Science 7 (1).

[2] Bengtson, J., Parrow, J., 2009. Formalising the pi-calculus using nominal logic. Logical Methods in Computer Science 5 (2).

[3] Bocchi, L., Wischik, L., 2004. A process calculus of atomic commit. Electronic Notes in Theoretical Computer Science 105 (0), 119 – 132, proceedings of the First International Workshop on Web Services and Formal Methods (WSFM 2004).

[4] Boreale, M., Fournet, C., Laneve, C., 1998. Bisimulations in the join-calculus. In: Programming Concepts and Methods PROCOMET âĂŹ98. IFIP âĂŤ The International Federation for Information Processing. Springer US, pp. 68–86.

[5] Borgström, J., Gutkovas, R., Parrow, J., Victor, B., Pohjola, J. Å., 2013. A sorted semantic framework for applied process calculi (extended abstract). In: Trustworthy Global Computing. pp. 103–118.

[6] Boudol, G., 1985. Notes on algebraic calculi of processes. In: Logics and Models of Concurrent Systems. Springer-Verlag New York, Inc., New York, NY, USA, pp. 261–303.

[7] Boudol, G., 1992. Asynchrony and the pi-calculus. Rapport de Recherche 1702.

[8] Busi, N., Gorrieri, R., Zavattaro, G., 2000. On the expressiveness of linda coordination primitives. Information and Computation 156 (1-2), 90–121.

[9] Carbone, M., Maffeis, S., May 2003. On the expressive power of polyadic synchronisation in π-calculus. Nordic Journal of Computing 10 (2), 70–98.

[10] Cardelli, L., Gordon, A. D., 1998. Mobile ambients. In: Foundations of Software Science and Computation Structures: First International Conference, FoSSaCS '98. pp. 140–155.

[11] Castagna, G., Nicola, R. D., Varacca, D., May 2008. Semantic subtyping for the pi-calculus. Theoretical Computer Science 398 (1-3), 217–242.

[12] De Nicola, R., Gorla, D., Pugliese, R., May 2006. On the expressive power of klaim-based calculi. Theoretical Computer Science 356 (3), 387–421.

[13] de Simone, R., 1985. Higher-level synchronising devices in Meije-SCCS. Theoretical Computer Science 37, 245–267.

[14] Ene, C., Muntean, T., 2001. A broadcast-based calculus for communicating systems. In: Parallel and Distributed Processing Symposium, International. Vol. 3. IEEE Computer Society, pp. 30149b–30149b.

[15] Fournet, C., Gonthier, G., ???? The reflexive cham and the join-calculus. In: Proceedings of the 23rd ACM Symposium on Principles of Programming Languages. ACM Press, pp. 372–385.

[16] Gelernter, D., 1985. Generative communication in Linda. ACM Transactions on Programming Languages and Systems 7 (1), 80–112.

[17] Given-Wilson, T., 2012. Concurrent Pattern Unification. PhD thesis, University of Technology, Sydney, Australia.

[18] Given-Wilson, T., 2014. An intensional concurrent faithful encoding of turing machines. In: Lanese, I., Lluch-Lafuente, A., Sokolova, A., Vieira, H. T. (Eds.), Proceedings 7th Interaction and Concurrency Experience, ICE 2014, Berlin, Germany, 6th June 2014. Vol. 166 of EPTCS. pp. 21–37.

[19] Given-Wilson, T., Sep. 2014. On the Expressiveness of Intensional Communication. In: Combined 21th International Workshop on Expressiveness in Concurrency and 11th Workshop on Structural Operational Semantics. Rome, Italie.

[20] Given-Wilson, T., Gorla, D., 2013. Pattern matching and bisimulation. In: Coordination Models and Languages. Vol. 7890 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 60–74.

[21] Given-Wilson, T., Gorla, D., Jay, B., 2010. Concurrent pattern calculus. In: Theoretical Computer Science. Vol. 323 of IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, pp. 244–258.

[22] Given-Wilson, T., Gorla, D., Jay, B., 2014. A Concurrent Pattern Calculus. Logical Methods in Computer Science 10 (3).

[23] Given-Wilson, T., Legay, A., Jun. 2015. On the Expressiveness of Joining. In: 8th Interaction and Concurrency Experience (ICE 2015). Grenoble, France.
URL https://hal.inria.fr/hal-01152456

[24] Gorla, D., 2008. Comparing communication primitives via their relative expressive power. Information and Computation 206 (8), 931–952.

[25] Gorla, D., 2010. A taxonomy of process calculi for distribution and mobility. Distributed Computing 23 (4), 273–299.

[26] Gorla, D., 2010. Towards a unified approach to encodability and separation results for process calculi. Information and Computation 208 (9), 1031–1053.

[27] Haack, C., Jeffrey, A., Aug. 2006. Pattern-matching spi-calculus. Information and Computation 204 (8), 1195–1263.

[28] Honda, K., Tokoro, M., 1991. An object calculus for asynchronous communication. In: ECOOP'91 European Conference on Object-Oriented Programming. Springer, pp. 133–147.

[29] Honda, K., Yoshida, N., 1995. On reduction-based process semantics. Theoretical Computer Science 152, 437–486.

[30] Lanese, I., Pérez, J. A., Sangiorgi, D., Schmitt, A., 2010. On the expressiveness of polyadic and synchronous communication in higher-order process calculi. In: Proceedings of International Colloquium on Automata, Languages and Programming (ICALP). Vol. 6199 of LNCS. Springer, pp. 442–453.

[31] Lanese, I., Vaz, C., Ferreira, C., 2010. On the expressive power of primitives for compensation handling. In: Proceedings of the 19th European Conference on Programming Languages and Systems. ESOP'10. Springer-Verlag, Berlin, Heidelberg, pp. 366–386.

[32] Milner, R., 1993. The polyadic π-calculus: A tutorial. In: Logic and Algebra of Specification. Vol. 94 of Series F. NATO ASI, Springer.

[33] Milner, R., Parrow, J., Walker, D., Sep. 1992. A calculus of mobile processes, I. Information and Computation 100 (1), 1–40.

[34] Milner, R., Parrow, J., Walker, D., Sep. 1992. A calculus of mobile processes, II. Information and Computation 100 (1), 41–77.

[35] Nicola, R. D., Ferrari, G. L., Pugliese, R., 1998. KLAIM: A kernel language for agents interaction and mobility. IEEE Transactions on Software Engineering 24 (5), 315–330.

[36] Nielsen, L., Yoshida, N., Honda, K., 2010. Multiparty symmetric sum types. In: Proceedings of the 17th International Workshop on Expressiveness in Concurrency (EXPRESS 2010). pp. 121–135.

[37] Palamidessi, C., Oct. 2003. Comparing the expressive power of the synchronous and asynchronous pi-calculi. Mathematical. Structures in Comp. Sci. 13 (5), 685–719.

[38] Parrow, J., Apr. 2008. Expressiveness of process algebras. Electronic Notes in Theoretical Computer Science 209, 173–186.

[39] Prasad, K. V., 1995. A calculus of broadcasting systems. Science of Computer Programming 25 (2), 285–327.

[40] Saraswat, V. A., Rinard, M., Panangaden, P., 1991. The semantic foundations of concurrent constraint programming. In: Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL '91. ACM, New York, NY, USA, pp. 333–352.

[41] Schmitt, A., Stefani, J., 2003. The m-calculus: a higher-order distributed process calculus. In: Aiken, A., Morrisett, G. (Eds.), Conference Record of POPL 2003: The 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, New Orleans, Louisisana, USA, January 15-17, 2003. ACM, pp. 50–61. URL http://dl.acm.org/citation.cfm?id=604131

[42] Urban, C., Berghofer, S., Norrish, M., 2007. BarendregtâĂŹs variable convention in rule inductions. In: Automated Deduction âĂŞ CADE-21. Vol. 4603 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 35–50.

[43] van Glabbeek, R. J., 2012. Musings on encodings and expressiveness. In: Proceedings of EXPRESS/SOS. Vol. 89 of EPTCS. pp. 81–98.